CHARACTERIZATION AND CONTROL OF NETWORK TRAFFIC: FROM HOURS TO NANOSECONDS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Chiun Lin Lim

January 2015

© 2015 Chiun Lin Lim
ALL RIGHTS RESERVED

CHARACTERIZATION AND CONTROL OF NETWORK TRAFFIC:
FROM HOURS TO NANOSECONDS

Chiun Lin Lim, Ph.D.
Cornell University 2015

This thesis covers the challenges faced by network operators and network users either individually or jointly on different timescales. In the first part of the thesis, we investigate the interaction between traffic engineering and TCP. In the network layer, a network operator directly controls the traffic via traffic engineering and indirectly influences the user offered traffic via feedback signals. In the transport layer, users send traffic into the network using the TCP protocol, which adjusts offered traffic according to the received feedback. We investigate how current traffic engineering practice interact with congestion control under the network utility maximization framework. We show that the current interaction is stable, increases network utility, but does not necessarily improve the traffic engineering objective. To jointly optimize the non-convex congestion control and multipath routing problem, we note the mismatch in incentive and take on a more holistic view using game theory. With change of variables, we obtain an equivalent convex optimization problem and with suitable modification of the feedback, we show that the interaction converges to the globally optimal solution of the equivalent convex problem for users running either primal or dual algorithms. We further show that the results hold even when traffic engineering is performed at any irregular intervals. More generally, we show via heterogeneous feedback the same optimality result for a mix of users running primal and dual algorithms.

In the second part of the thesis, we first lay down the framework of network traffic dynamics by specifying the governing equations for the time evolutions, dynamics and associated delays of the network elements. We next specialize the framework to study how a centrally controlled network could reconfigure its routing as quickly as possible while not incurring any congestion. As switches may update at different times and update to different traffic flows take different times to propagate through the network, transient congestion could occur when links contain a mix of traffic flows following old and new routing configurations. Using propagation delay information from the framework and incorporating timing uncertainty, we figure out which congestion scenario could occur and how long it would take any update to properly propagate through the network. We formulate a mixed-integer linear program to find fast congestion-free routing reconfiguration using timing information. We explore how we could fasten the update process as we do not have to wait for an update to fully propagate through the network. For heavily congested network, we show a fast update solution by trading off the minimal amount of traffic demand. Experiments on Mininet verify our approach and show that it outperforms prior method with no timing information.

In the final part, we investigate router's inherent variation on packet processing time and its effect on interpacket delay and packet clustering. We propose a simple pipeline model incorporating the inherent variation, and two metrics, one to measure packet clustering and one to quantify inherent variation. To isolate the effect of the inherent variation, we begin our analysis with no cross traffic and step through setups where the input streams have different data rate, packet size and go through different number of hops. We show that a homogeneous input stream with a sufficiently large interpacket gap will emerge at the

router's output with interpacket delays that are negatively correlated with adjacent values and have symmetrical distributions. For an input with smaller interpacket gap, the change in packet clustering is smaller while for a more clustered input, the change is also smaller and could actually go negative. We generalize our results by adding cross traffic and show how the understanding gained could be applied to engineer traffic with minimal jitter. The model analysis is validated with experiments using SoNIC, a highly precise instrument providing real-time access to the physical layer.

BIOGRAPHICAL SKETCH

Chiun Lin Lim earned his Bachelor of Science degree in Electrical and Computer Engineering from Cornell University in 2008. He received his Master of Engineering degree in Electrical and Computer Engineering from Cornell University in 2009, then joined the doctoral degree program in Electrical and Computer Engineering that same year. He is a member of the Networks Group, led by Prof. A. Kevin Tang. He was awarded a scholarship from Malaysian's Public Service Department and a recipient of Jacobs Fellowship. While pursuing his degree, he has worked as a teaching assistant and Eta Kappa Nu tutor in the school of Electrical and Computer Engineering. His research interests include networks, optimization and game theory.

This thesis is dedicated to my parents.

ACKNOWLEDGEMENTS

First, I would like to express my deepest gratitude to my advisor Prof. A. Kevin Tang for his support, patience, motivating words and insight throughout my Ph.D. study. He has granted me much academic freedom to explore my interests and at the same time provided guidance throughout the research and writing of this thesis. I would also like to thank the other members of my graduate committee for being a critical part of my Ph.D. study. To Prof. Hakim Weatherspoon, thanks for providing me with the opportunity to do a collaborative study using SoNIC. To Prof. Salman Avestimehr, thanks for the insightful comments and questions.

I also owe my heartfelt gratitude to the various people whom I collaborate with. To Ki Suh Lee and Han Wang, thanks for enlightening me on the intricacies of SoNIC, for the various fruitful discussion and for accommodating various requests for experimental verifications. To Ning Wu and Shih-Hao Tseng, thanks for the help and discussion while we explore congestion-free routing reconfiguration.

My sincere thanks also goes to the various professors that have influenced me prior to my graduate study. To Dr. David F. Delchamps, thanks for the opportunity to work on an exciting Masters of Engineering project on evolutionary game theory. To Prof. Michael J. Todd, thanks for piquing my interest in game theory. To Prof. Steven H. Strogatz, thanks for the wonderful lectures on nonlinear dynamics.

I would like to thank my many friends that were part of the process in one way or another. To my lifelong friends from Malaysia, Siew Han Sim, Cheng Yong Teo, Sian Yau Oh and Tser Chin Tan, thanks for always being there. To my fellow office mates, Dr. Enrique Mallada, Dr. Nithin Michael, Dr. Alireza

Vahid, Ibrahim Issa, and Raphael Louca, thanks for the discussions and fun that we had together. To the Malaysians in Cornell, Yong Sheng Khoo, Byron Singh, Benjamin Tang, Jon San Ng, Xiao Rui Loo, Wei Jen Lim, Brandon Lim, Wei Kiat Chen, Tze Jian Chear, Jun Hui Erh, Esther Lim, Ken Hui Fu, Iswari Nallisamy, Si Ning Yeoh, Bryan Chong, Guo Jie Chin, Darren Voon, Tzer Han Tan, Ai Hui Chew, Kai Lin Tan, Carmen Loh, Jian Cheng Wong, Jie Yuen Ong, Chern Wei Bee, Khai Zhi Sim, Sarah Tan, Jun Le Goh, Pauline Sho, and Bryan Tam, thanks for providing a community to be at home while halfway across the globe from home. To my friends throughout my Cornell years, Ellen Lee, Ziyan Huang, Wenrong Lim, Chin Ning Ho, Hain-Lee Hsueh, Moonjoo Cho, Alexander Hartoto, and Jacky Wang, thanks for all the fun times and precious memories.

Last but not least, I would like to thank my family: my father Ah Chuan Lim, my mother Imm Gan, my brother Eu Tak Lim and my sister Jiun Jen Lim for their unconditional love and support.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Traffic engineering in communication networks deals with the application of engineering techniques to move, modify or shape traffic in the network to achieve various traffic-level objectives. The problem exists as far back as the early days of ARPANET, the predecessor of the Internet, where network operators were faced with the optimal routing problem [48, 42]. This is a resource allocation problem where network operators have to choose how traffic are routed through a network with link capacity constraint. The problem is of primary concern to network operators wanting to efficiently utilize the limited resources of a network. However, the problem assumes a static network condition and does not take into account either evolving network conditions arising from changing traffic demand or even from the change in routing itself.

At the same time, the Internet has evolved much since and with it, we have an ever increasing number of applications, each with its own diverse demands and objectives. As such, the realm of traffic engineering is no longer restricted to just blind optimization of the network operator objectives, but also with additional consideration to the requirements of applications. Multimedia applications require the packet stream to have low jitter, websites desire their pages to have low response times and online game needs a stable and bounded delay environment to prevent desynchronization between the client and the server. Incorporating application requirements, despite the fundamental differences and the potential conflict in objective, is a major challenge in traffic engineering.

1.1 Traffic Engineering

Traffic engineering is a multi-step process. The first step involves the network operator estimating the demand of the traffic flows into the network. The estimated values are the aggregated average values and the assumption here is that the demand of the traffic flow is held relatively constant around the estimated values before the next cycle of traffic engineering. Estimation are done by directly measuring the incoming traffic over a fixed period of time or by extrapolating from available historical traffic information.

Next, the obtained demand values are used as input to an optimal routing problem [48, 42], also well-known as the multi-commodity flow problem [23, 45]. In the optimal routing problem, the network operator, which controls a network of routers and links, has to satisfy the demand of traffic flows entering and exiting the network by choosing their routes through the network. At the same time, the routes chosen have to satisfy capacity constraints and optimize system-wide objectives such as delay or utilization. Solving the problem gives an optimal routing configuration.

The final step requires reconfiguring the current network state and routes to follow the optimal routing configuration. This is the point where routing protocols come in. Network operators reconfigure by tuning the parameters of these routing protocols. The complete process of traffic engineering is summarized in Figure 1.1.

The multi-commodity flow problem is well-studied and our focus is instead on the problems and challenges stemming from the initial and final step of the process. For the initial step, we are interested in knowing what would happen

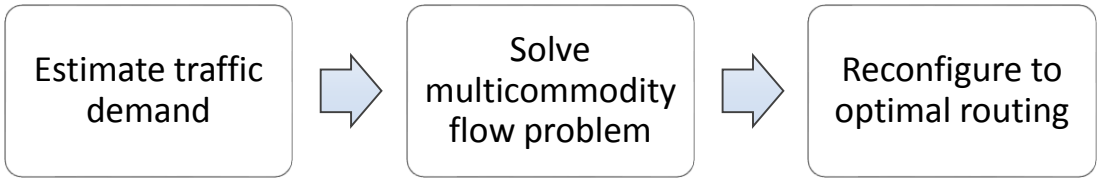


Figure 1.1: Quick view of the multi-step process of traffic engineering

when the constant traffic assumption is relaxed by having an incoming elastic traffic that varies according to the network condition along its path. We investigate how the elastic traffic interacts with traffic engineering and we focus on the long-term stability and optimality issues resulting from the dynamics of the interaction. The key questions that we would like to answer are:

- (*Stability*) Does the interaction converge to a fixed operating point where the incoming traffic does not change its demand and the network operator does not reconfigure the network routes?
- (*Optimality*) If the interaction is stable, is it optimal from the perspective of the user sending the traffic, the network operator or the system as a whole? If not, what could the network operator do to induce optimality?

For the reconfiguration step, there are three major challenges, namely *optimality*, *consistency* and *swiftness*. An optimal reconfiguration ensures that at the end of the process, the routing configuration that is in place is indeed the optimal routing configuration. Under a distributed setting, the predominant link-state routing protocols are Open Shortest Path First (OSPF) [79] and Intermediate System to Intermediate System (IS-IS) [22]. These protocols work by having each router having a full view of the topology of the network. Each link of the network is associated with a link weight. To figure out the route to take to a destination router for any incoming traffic flow, a router runs Dijkstra's al-

gorithm on the link weights and the resulting shortest path is the route that will be taken. To reconfigure the routes, network operators do it indirectly by tuning the link weights of each link. However, it turns out that setting the right link-weight is NP-hard, and sometimes even the best link-weight could have significant deviations from the optimal configuration [39, 37].

Alternatively, we have routing protocols that work by encoding the full source-destination route such as Multiprotocol Label Switching (MPLS) [74]. Instead of the routers choosing the route to take to different destinations for each incoming flow, the routes or tunnels are chosen at the ingress router for each source-destination pair. Similar to link-state routing protocols, the network operator reconfigures indirectly by tuning the link weight. The ingress router then uses to the link weights to solve a constrained shortest path problem. As the process is done in a distributed manner by each ingress router, there is no guarantee that the final routing configuration is the desired optimal configuration [84, 106].

The primary factors contributing to the lack of optimality are the lack of direct control by the network operator and the distributed nature of the routing protocol. As such, in recent years, we have an emerging architecture known as Software-Defined Networking (SDN) that is directly programmable and centrally managed. The extra control is achieved by allowing network operator to directly determine how traffic is routed instead of the indirect method as is done in traditional routing protocols. For instance, OpenFlow [6], which enables the SDN architecture, allows the network operator to directly modify the forwarding tables of each router in the network. As such, optimality could be easily guaranteed with a centralized controller directly setting the optimal routing configuration.

With optimality issues easily handled by the SDN architecture, our focus is on the last two challenges. The second challenge is concerned with the *transient* stages while reconfiguring to the optimal configuration and this is where consistency requirement comes in. A consistent update [88, 56] ensures certain network properties of interest such as in-order delivery, loop-freedom [103] or capacity constraint [51, 102] are satisfied during all transient stages of the routing reconfiguration. When implemented incorrectly, an inconsistent update could be a very costly exercise to the network operator. The Amazon EC2 and RDS service disruption in April 21, 2011, is one such example [8]. The reconfiguration begins with the intention to upgrade the capacity of the network and this requires some of the network traffic to be shifted around. The network traffic was incorrectly routed to a router which could not handle the incoming traffic. The route was corrected not long after, but the mistake caused a cascading effect on the services which took days to fully recover.

The final challenge is to perform the reconfiguration in the least amount of time possible. As mentioned at the beginning of the section, the optimal configuration is based on the assumption that the demand of the traffic flow is held relatively constant around the estimated values. As such, the optimal configuration is only valid while the estimated values are reasonable approximation of the actual traffic demand. A swift update ensures the final routing configuration does not become obsolete due to fast changing network conditions and cause lower network utilization.

1.2 User Requirements

Application (User[1]) requirements could be broadly classified into three categories: delay, throughput and packet positioning. Each application has varying degrees of requirements on these three categories. Applications sensitive to delay requires their packets to get to the destination within a set period of time. The delay requirement typically arises from application requiring interactivity, such as teleconferencing or online gaming, where delayed responses would severely hamper the user experience. Throughput requirements are about the number of packets transferred within one round-trip time and comes in several variants. Applications could either require a set amount of bandwidth, such as video streaming; could desire as much bandwidth as possible, such as file transfer; or could desire packet transfer with bounded packet losses as in wireless networks. Lastly, applications could have requirements on how the packets should arrive at the destination. It could be desirable for the packets to arrive in-order, or for the packets to arrive with minimal jitter as in multimedia applications.

The task of the application is to control how the packets should be sent in order to satisfy these requirements. Depending on the requirement, the interaction of the application with the network differs greatly. Applications that care about throughput and delay typically have an elastic traffic demand that varies according to the detected network condition. The traffic demand is higher when the network condition is good and vice versa. Such action could either conflict with the network operator's objective as sending more traffic into the network would necessarily degrade the network condition or help the network opera-

[1]We will use applications and users interchangeably

tor by curbing the traffic demand to reduce the network load. As mentioned in Chapter 1.1, the interaction of these applications with the network operator's traffic engineering efforts creates an interesting dynamics and we are interested in its long-term stability and optimality issues.

Packet positioning requirements operate at a finer timescale. While delay, throughput, and traffic engineering are more about aggregate and average flow-level requirements, packet positioning requirements are packet-level requirements. As such, we care more about its interaction with network elements and other cross traffic. Thus, a fundamental understanding of how network elements operate and how they affect the packets passing through them are crucial to designing control schemes that satisfy the requirements of these application. In particular, we focus on how randomness inherent in a router affects jitter.

1.3 Timescale

We have briefly gone through the perspectives and challenges of the network operator and the applications in Chapter 1.1 and 1.2. We are still missing one piece to complete the introductory picture though, and that is the timescale differences of the problems. Each of the problems that we just discuss actually occupy a different timescale (summarized in Table 1.1).

At the smallest timescale we have the packet-level spanning nanoseconds to microseconds. This is the timescale with a similar order as the time it takes to

Table 1.1: Timescale differences of packet-level, flow-level and network-level

Level	Time (seconds)	Related phenomena	Problem
Packet	$10^{-9} - 10^{-4}$	Transmission time of a packet, jitter, bandwidth estimation	Inherent variation of a router and its effect on packet clustering
Flow	$10^{-3} - 10^{0}$	Round-trip time, window flow control, routing updates	Consistent and swift routing reconfiguration
Network	$10^{1} - 10^{3}$	Multiple traffic engineering updates, long-term network behavior	Stability and optimality of network-user Interaction

transmit a packet of size 64 to 1500 bytes onto a 10 Gbps link:

$$\begin{aligned} (64 \times 8 \text{ bits}) \,/\, \left(10 \times 10^9 \text{ bits/s}\right) &= 51.2 \text{ nanoseconds} \\ (1500 \times 8 \text{ bits}) \,/\, \left(10 \times 10^9 \text{ bits/s}\right) &= 1.2 \text{ microseconds} \end{aligned}$$

This is the timescale of interest for packet positioning requirements. Moving up we have the flow-level spanning milliseconds to seconds. This is the timescale of round-trip time for physical distances spanning hundreds to tens of thousands of kilometers:[2]

$$\begin{aligned} (2 \times 100 \text{ km}) \,/\, \left(2 \times 10^5 \text{ km/s}\right) &= 1 \text{ milliseconds} \\ (2 \times 10000 \text{ km}) \,/\, \left(2 \times 10^5 \text{ km/s}\right) &= 100 \text{ milliseconds} \end{aligned}$$

This is the timescale where network operators deal with aggregate traffic flows and perform routing reconfigurations. Finally, we have the network level spanning minutes to hours. Depending on the nature of the incoming traffic, network operators typically have an update period of seconds to minutes to carry out traffic engineering. This is the timescale where we investigate the interaction and long-term behavior between traffic engineering and elastic input traffic as they evolve over multiple traffic engineering updates.

[2] Assuming signals are traveling in fiber optic at a speed of 2×10^5 km/s

1.4 Contributions

We have introduced the various challenges faced by network operators, the applications and their interactions. This thesis is divided into three parts and each part of the thesis will deal with one or more of the challenges. In the first part, we look at the interaction between traffic engineering and users. We show how the users and network operator, despite the conflicting objectives, could work together to overcome the challenges and achieve social-wide objectives. In the second part, we define a framework for network traffic dynamics before we begin to explore consistent and swift updates for network operators under a Software Defined Networking architecture. Our focus is on congestion-free routing reconfiguration and we leverage the explicit timing information incorporated to achieve a faster reconfiguration than is previously possible. Finally, in the last part of the thesis, we investigate an unexpected source of uncertainty introduced by routing elements in the network. The study is based on a previously observed but unexplained phenomena whereby packets passing through a router with no cross traffic experiences variation in the transition time. The work is of particular relevance to applications requiring packet positioning such as bandwidth estimation, covert timing channels, and multimedia applications.

1.4.1 Network-User Interaction

The work here could be viewed from either the perspective of the network operator or the users. The theoretical framework underlying the operations of the network operator is traffic engineering as introduced in Chapter 1.1 while for the users, the underlying theoretical framework is network utility maximiza-

tion (NUM). Our main perspective is from the traffic engineering standpoint, and our main contribution is the relaxation of the constant traffic demand assumption to an elastic traffic following the dynamics of NUM. Our focus on traffic engineering separates us from prior network-user interaction works with a NUM standpoint [58, 47, 46, 106, 82] in three ways:

- The network operator controls the routing as opposed to source routing.
- The user does not have full knowledge of the network but instead only has limited information on the network path that it uses in the form of feedback, which is set by the network operator.
- There is a timescale difference between user updates and traffic engineering updates as opposed to requiring synchronous user and routing updates.

These differences introduce additional restrictions to the problem, but we are able to show that

- In general, the interaction between traffic engineering and NUM converges to a stable solution that is suboptimal with respect to a social objective function.
- Achieving social optimality is a non-convex problem, but we manage to find a transform that turns it into an equivalent convex problem.
- By borrowing the tools of game theory, refining the model and setting the appropriate feedback, we show that despite the timescale differences, the network-user interaction could achieve a stable and socially optimal solution.

- The stability and optimality results could be generalized to a network with heterogeneous users.

1.4.2 Traffic Engineering Dynamics

In this part of the thesis, we first lay down the general framework of network traffic dynamics. The framework deals with flow-level dynamics by explicitly characterizing all the delays in the network and the time evolution of the state of network elements. We next specialize the framework to the study fast congestion-free routing reconfiguration by adding in timing uncertainty information. The delay information from the network traffic dynamics framework, together with the added timing uncertainties, provide many benefits over prior work [51]:

- We can figure out how long it would take an updated routing configuration to fully propagate through any links and through the whole network. We no longer have to wait an arbitrarily long period of time before we can execute the next update.
- We no longer have to assume worst-case scenarios in terms of which routing update would arrive at a link. We are less constrained and thus are able to find faster updates.
- Further speed-up could be obtained as the subsequent update could sometimes be performed without violating capacity constraints even before the current update has fully propagated through the network. Such a speed-up is not possible without timing information.

- Even when a congestion-free solution does not exist, we show how routing reconfiguration could be done with a minimal loss in traffic demand.
- The model is robust to measurement uncertainty in the delays. The more precise the measurement, the faster the reconfiguration.

1.4.3 Inherent Variation in Routers

In the absence of external factors such as cross traffic, a packet passing through a router is expected to have a deterministic transition time. The experiments performed in [43] show that this is not the case as routers have some inherent variation causing some variability in the transition time. In some cases, the variability is sufficient to induce bursty traffic wherein packets cluster together. The observed phenomena was left unexplained though, and we build from there:

- We propose a simple device-independent router model incorporating inherent variation. The model admits a pipeline structure. Using the model, we make several predictions on the properties of the interpacket delay at the router's output. We predict that adjacent interpacket delays are negatively correlated (Theorem 4.2) and the histogram of the interpacket delay is symmetrical in shape (Theorem 4.3).
- We propose a metric to quantify packet clustering. We show how packet clustering in terms of this metric changes as it passes through one or multiple routers (Theorem 4.6 and Section 4.6.2). We also develop conditions on when the output is less clustered than the input (Corollary 4.7).
- We generalize our results by incorporating cross traffic (Lemma 4.9 and Theorem 4.11).

- We are able to provide qualitative and quantitative explanations on the observed phenomena of [43]. We also verify all the model results with repeatable experiments using SoNIC [62]
- We investigate the impact of packet sizes, clock drift, and forwarding table lookup on the variability of transition time.
- We show how we could apply the understanding derived from the modeling to minimize jitter.

CHAPTER 2

NETWORK-USER INTERACTION OF TRAFFIC ENGINEERING (MINUTES - HOURS)

2.1 Background

The focus here is on users concerned about throughput while for the network operator, its about optimality such as total delay in the network. The objective of the applications is in conflict with those of the network operator: sending more traffic into the network would necessarily worsen the network condition. From the perspective of a layered network architecture, our work in this chapter deals with the interaction of transport layer (congestion control) and network layer (routing). Due to potential confusion with the application layer, we refer to applications as users instead. The congestion control optimization framework, i.e. network utility maximization (NUM) [97] has been developed assuming the traffic flow from each user follows a single static path. Investigations on relaxation of the fixed routing assumption has been around for quite some time. The joint congestion control and multipath routing problem is already present in Kelly's seminal work [58]. There, rate variables are defined for all possible paths from source to destination, and each user has to determine all these rates to maximize utility. More recent works have shifted the routing decisions to the nodes. Each edge router either separately chooses a route to minimize congestion cost [106] or determines, for each destination, the load to place on all possible paths [46]; or all routers control the per-destination split ratios on outgoing links [82]. The paper that is closest in essence to ours is [47] though our stability and optimality results are not limited to the ring topology and our

timescale separation model is more general. Many other related papers could be found in [27] but in general, investigations in this direction will encounter the following limitations:

1. **Knowledge of the network topology.** Network users could obtain information about the paths used via feedback but beyond that have limited to no knowledge of the network topology and possible paths to destination.
2. **Timescale separation.** Congestion control converges to equilibrium in the order of round-trip time. Routing is generally updated by the network operator on a longer timescale. This timescale mismatch means that algorithms updating congestion control and routing update simultaneously are usually impractical.
3. **Network ownership.** Probably the most important limitation is that the network infrastructure is owned and controlled by the network operator. A network operator routes the traffic according to his own incentives such as congestion minimization or revenue maximization. Such incentives are typically not aligned with the utility maximization of users.

Across the aisle, the traffic engineering community has much fewer works dealing with the interaction. The hardness of the link-weight setting problem [39] and the lack of a solid analytical framework compared to NUM has limited the cross-layer works, for instance [36], to be simulation in nature. There is a vast literature on traffic engineering on networks, however, if we do not limit ourselves to communication networks. Research on transportation traffic has been around for decades [73]. Beside the convex optimization tools [19] employed in NUM, another important theoretical tool here is game theory [44] where agents are modeled to behave selfishly. Under the game the-

oretical framework, each agent acts independently to choose a minimum-cost path, given the congestion costs caused by the actions of all other players. The problem is usually modeled as a non-cooperative congestion game, which is guaranteed to have a pure-strategy Nash equilibrium [91]. It is well-known that in general Nash equilibria are not necessarily socially optimal. How inefficient any Nash equilibrium can be has been characterized with the concept of Price of Anarchy (PoA) in [93, 92]. PoA is the ratio between the social value of the socially optimally solution and that of the worst Nash equilibrium. These works have been done with a constant offered traffic assumption and extensions to elastic traffic could be found in [28, 25, 54]. Viewed from the communication network framework, investigations from a game theoretical standpoint will run into the aforementioned limitations — knowledge of the network topology and timescale separation — as well as the following:

- **User behavior.** The behavior of a network user is governed by the congestion control algorithm, which is not necessarily selfish and could not be controlled by the network operator.

Our objective is to take on the role of the network operator and design the appropriate traffic engineering steps to achieve cross-layer optimality while conforming to all four limitations. We start by modeling the interaction of traffic engineering and NUM as a two-stage iterative process in Section 2.2 and explain how currently feedback, traffic engineering, and NUM interact with each other. In Section 2.3, we find that the process converges, improves network utility, but does not guarantee improvement in the traffic engineering's objective. We thus relax the capacity constraint and propose a modification under a game theory framework in Section 2.4, albeit *users are not necessarily selfish but instead*

act according to NUM. In Section 2.5, we formulate a potential game perspective for the primal algorithm of NUM and show that the primal algorithm converges to the socially optimal solution. We next relax our timescale separation assumption and show that the same convergence holds even when traffic engineering is performed at any irregular intervals. We extend to the dual algorithm for NUM in Section 2.6 and show that with proper modification of the feedback signal, the convergence results also hold here. In Section 2.7, we relax the homogeneous user assumption by allowing heterogeneous users running primal or dual algorithms and prove the same optimality result.

2.2 Model formulation

We consider a network controlled by a network operator, whom we refer interchangeably as the traffic engineer in this chapter. A set of N users, each representing a particular source-destination pair, sends traffic into the network. We assume users have infinite backlog and any user traffic could be split at any ratio across all possible paths from source to destination. An illustrative figure of the model is shown in Figure 2.1.

Before proceeding further, for clarity, we explain the notational conventions adopted throughout this chapter. Uppercase letters denote matrices, e.g. H, Q, R, or sets, e.g. L, N[1], or constants, e.g. K, or utilities (or costs), e.g. U, Φ. Lowercase letter i represents user, j or k represents path, l represents link and other letters such as x, f, q denote vectors. Superscript denotes element of a vector, e.g. x^i or columns of a matrix, e.g. H^i associated with user i. Similarly, subscript

[1]We will abuse notation by using L and N to denote both sets and their cardinality.

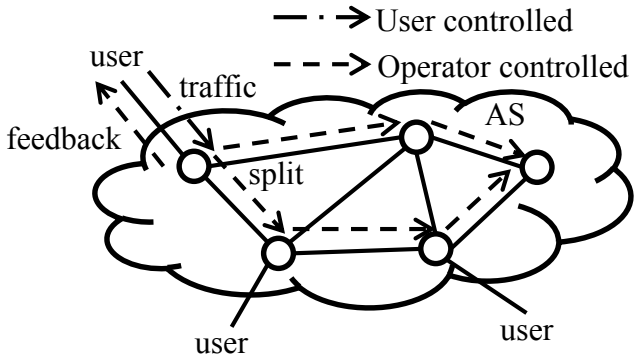


Figure 2.1: User controls input traffic while network operator owns the network and controls feedback and traffic split at routers

Table 2.1: Table of Notations

Symbol	Description
N	Set/Cardinality of users
L	Set/Cardinality of directed links
H	Topology matrix
Q	Traffic split matrix
R	Routing matrix
f_l	Total traffic flow through link l
$\Phi_l(f_l)$	Cost function of link l
$\phi_l(f_l)$	Price function of link l
c_l	Capacity of link l
x^i	Offered traffic of user i
$U^i(x^i)$	Utility function of user i
$\Lambda(\cdot)$	Traffic engineering objective function

on a vector or matrix, e.g. f_l, q_j^i denotes association with either link l or path j. A bar, e.g. $\bar{x}^i$, means the variable is an optimal solution. The bold number $\mathbf{1}$ is the vector of all ones while e_l or e_j^i is a unit vector for the corresponding element. Whenever a lemma or theorem is stated, it is implicit that all the assumptions that have been stated in the paper thus far holds.

We model the network as a graph with a set L of directed links. The links have finite capacities $c = \{c_l | l \in L\}$ and each link l is associated with a link-

cost function $\Phi_l(f_l)$ that is convex and increasing in f_l, the traffic flow through link l. Examples of $\Phi_l(\cdot)$ are link utilization f_l/c_l and link delay $f_l/(c_l - f_l)$ as given by the $M/M/1$ formula [104].

Each user $i \in N$ is associated with K^i acyclic paths, which are represented by a $L \times K^i$ $0-1$ matrix H^i where

$$H^i_{lj} = \begin{cases} 1, & \text{if user } i \text{ uses link } l \text{ in path } j \\ 0, & \text{otherwise} \end{cases}$$

The matrix H^i does not necessarily contain all the possible paths from source to destination but instead could just be a subset of the paths. Let $K = \sum_i K^i$ and define the $L \times K$ matrix as

$$H = \begin{bmatrix} H^1 & \dots & H^N \end{bmatrix}$$

A split ratio is specified over the K^i paths by the $K^i \times 1$ vector q^i where q^i_j represents the fraction of i's flow on path j such that

$$q^i_j \geq 0, \qquad \mathbf{1}^T q^i = 1$$

Collect the vectors q^i, $i = 1, \dots, N$ into a $K \times N$ block diagonal matrix $Q = \text{diag}\left(q^1, \dots, q^N\right)$. We represent the set of all possible traffic split matrices as

$$\mathcal{Q} = \left\{ Q | Q = \text{diag}\left(q^1, \dots, q^N\right) \in [0,1]^{K \times N}, \mathbf{1}^T q^i = 1 \right\}$$

The matrix H defines the topology of the network while Q represents how the traffic of each user is split over the available paths. Their product is a $L \times N$ routing matrix $R = HQ$ with its R_{li} entry giving the fraction of i's traffic at each link l.

We model the current interaction between the users and the traffic engineer by combining the models of NUM and traffic engineering into an iterative two-

stage process. We will not achieve optimality with this initial model, but analyzing it will give some important insights into the two-stage process and on how we could refine the model to achieve optimality. First, the traffic engineer, with knowledge of the offered traffic $x = \left\{x^i | i \in N\right\}$, solves a traffic engineering problem

$$\begin{aligned} (TE) \min_{Q \in \mathcal{Q}} \quad & \Lambda\left(HQx\right) \\ \text{s.t.} \quad & HQx = f \leq c \end{aligned}$$

where $f = \{f_l | l \in L\}$ represents the load on all the links. Examples of the objective function Λ are maximum utilization, $\max_l f_l / c_l$, and for additive cost function, $\sum_l \Phi_l\left(e_l HQx\right) = \sum_l \Phi_l\left(f_l\right)$, in which case we could be minimizing total delay.

Additionally, the traffic engineer sets a congestion price for all links. The price is represented by a price function $\phi = \{\phi_l\left(f_l\right) : l \in L\}$ that is related to the link cost function Φ. For each path used, the edge router of user i receives the sum of all link prices along the path as feedback. Thus, the price information received by the edge router is the vector $\left(H^i\right)^T \phi \in \mathbb{R}^{K^i}$. The edge router then calculates the expected congestion price $\phi^T H^i q^i = \left(HQe_i\right)^T \phi$ and passes it to the user as a feedback value. Users update their offered traffic according to the feedback value.

Solving (TE) gives a new traffic split matrix Q, and users update their offered traffic as the feedback values have changed. The update is governed by the NUM framework where users solve the following problem: Given Q,

$$\begin{aligned} (\text{NUM}) \max_{x \geq 0} \quad & \sum_{i \in N} U^i\left(x^i\right) \\ \text{s.t.} \quad & Rx \leq c \end{aligned}$$

where $U^i\left(x^i\right)$ is a non-negative, increasing, and strictly concave utility function. The NUM framework comes with two flavors: primal and dual. Under the primal formulation, the capacity constraint is relaxed by adding a barrier function to the objective

$$\text{(NUM-P)} \qquad \max_{x \geq 0} \quad \sum_{i \in N} U^i\left(x^i\right) - \sum_{l \in L} B_l\left(e_l^T R x\right)$$

By proper choice of a convex barrier function, $B_l(\cdot)$, the optimal solution of (NUM) could be approximated by solving (NUM-P) [12]. For single path, i.e. when Q is restricted to be a $0-1$ matrix, the primal algorithm has the users and price function update as follows [97]:

$$\dot{x^i} = \kappa^i\left(x^i\right)\left[\left(U^i\right)'\left(x^i\right) - \left(Re^i\right)^T \phi\right] \tag{2.1}$$

$$\phi_l = B_l' \tag{2.2}$$

where $\kappa^i(\cdot)$ is non-negative, increasing and continuous. For the dual formulation, (NUM) is solved directly and the dual algorithm has the users and price function update as follows [69]:

$$x^i = \left(U^i\right)'^{-1}\left(\left(Re^i\right)^T \phi\right) \tag{2.3}$$

$$\dot{\phi}_l = \begin{cases} h_l\left(f_l - c_l\right) & , \ \phi_l > 0 \\ h_l \max\left\{f_l - c_l, 0\right\} & , \ \phi_l = 0 \end{cases} \tag{2.4}$$

where h_l is a positive constant.

We assume that the update on Q is instantaneous. We also assume for now that the users converge to the optimal solution of (NUM) before the traffic engineer updates Q again. Due to the timescale separation limitation that we mentioned earlier, this is a reasonable assumption to make, though we will relax

it in the later section to cover not only the 3 timescale separation models mentioned in [47], but also allowing Q to be updated at any irregular intervals. With this assumption, the iterative two-stage process has the traffic engineer and the users taking turns solving their respective optimization problems:

$$Q(t+1) = \arg \min_{Q \in \mathcal{Q}: HQx(t) \leq c} \Lambda(HQx(t)) \tag{2.5}$$

$$x(t+1) = \arg \max_{x \geq 0: HQ(t+1)x \leq c} \sum_{i \in N} U^i\left(x^i\right) \tag{2.6}$$

When there are more than one optimal solution to (2.5), we adopt the convention that if $Q(t)$ is an optimal solution, then we choose $Q(t+1) = Q(t)$. If not, an optimal solution is chosen at random. This convention of always choosing the current solution if optimal holds as well for (2.6) and in the later optimization problems.

We consider an example to familiarize with the notation and to illustrate the iterative two-stage process.

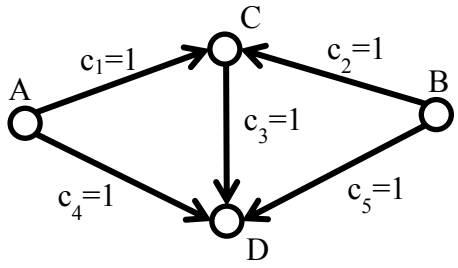


Figure 2.2: An example with link-cost function $\Phi_l(f_l) = f_l^2/2$

Example 2.1. (Illustrative example) Consider the network shown in Figure 2.2. The cost function for each link l is $\Phi_l(f_l) = f_l^2/2$. There are two users with the same strictly concave utility function $U(\cdot)$. The first user sends traffic from A to D via $A \to C \to D$ (path 1) and $A \to D$ (path 2) while the second user sends

traffic from B to D via $B \to C \to D$ (path 1) and $B \to D$ (path 2).

$$H_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad H_2 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad Q = \begin{bmatrix} q_1^1 & 0 \\ q_2^1 & 0 \\ 0 & q_2^1 \\ 0 & q_2^2 \end{bmatrix}$$

Suppose initially, the first user offers $x^1(t=0) = 1/2$ unit of traffic and the second user offers $x^2(t=0) = 1/4$ unit of traffic. Both offered traffic are initially set by the traffic engineer to travel on path 2, i.e. $q_2^1(0) = q_2^2(0) = 1$.

At time $t = 1$, to solve (TE), we utilize the well-known fact stating that for multi-commodity flow problem, if capacity constraints do not come into play, then at the optimal solution, for each user, the first derivative length is the same for all paths with positive rates [13]. This translates into the conditions

$$\begin{aligned} x_1^1 + \left(x_1^1 + x_1^2\right) &= x^1(0) - x_1^1 \\ x_1^2 + \left(x_1^1 + x_1^2\right) &= x^2(0) - x_1^2 \end{aligned}$$

where x_j^i is the optimal offered traffic for user i on path j for (TE). Solving the linear system and rescaling the results give the optimal split ratios as

$$\begin{aligned} q_1^1(1) = \frac{3x^1(0) - x^2(0)}{8x^1(0)} &< \frac{5x^1(0) + x^2(0)}{8x^1(0)} = q_2^1(1) \\ q_1^2(1) = \frac{3x^2(0) - x^1(0)}{8x^2(0)} &< \frac{5x^2(0) + x^1(0)}{8x^2(0)} = q_2^2(1) \end{aligned}$$

The two users now adjust their offered traffic according to the new traffic split. Regardless of the utility function and feedback, under (NUM), optimality is achieved when the users each increases the offered traffic until it hits the capacity constraint on the second path. For user 1, $1/2$ unit of traffic is sent along

path 2 implying a total offered traffic of

$$x^1(1) = \frac{1/2}{q_2^1(1)} = \frac{4x^1(0)}{5x^1(0) + x^2(0)} = \frac{8}{11}$$

Similarly for user 2, $1/4$ unit of traffic is sent along path 2 implying a total offered traffic of

$$x^2(1) = \frac{1/4}{q_2^2(1)} = \frac{2x^2(0)}{5x^2(0) + x^1(0)} = \frac{2}{7}$$

At $t = 2$, capacity constraint comes into play for user 2 as $x_1^2(1) + \left(x_1^1(1) + x_1^2(1)\right) = 23/77 > 1/4$ and thus the optimal solution for (TE) for user 2 remains unchanged, i.e. $q^2(2) = q^2(1)$. For user 1, capacity constraint does not come into play as long as $x_1^1(t-1) < 13/56$ or equivalently $x^1(t-1) < 41/56$. The condition is satisfied and so going through the same line of reasoning as before gives the relationship

$$\frac{41}{56} > x^1(t) = \frac{42x^1(t-1)}{56x^1(t-1) + 1} > x^1(t-1) \tag{2.7}$$

The left inequality holds since $42x^1(t-1) < 41x^1(t-1) + 41/56$ while the right inequality is obtained by applying $x^1(t-1) < 41/56$ on the denominator.

For $t \geq 3$, one could check that $q^2(t) = q^2(1)$ and the chain of relationships in (2.7) hold. As $t \to \infty$, the iterative two-stage process converges to a total network utility of $U(41/56) + U(2/7) > U(1/2) + U(1/4)$ and a traffic engineering objective of $5/32 + 199/3136 > 5/32$. Before proceeding, remember the three key points listed below. We will hammer home the last two points with further examples.

- The two-stage iterative process converges (Theorem 2.1).
- The network utility strictly increases while the traffic engineer is worse off after each iteration (Example 2.2).

Table 2.2: A quick view of the interaction between the users and the traffic engineer in the most general setup

Layer	Transport	Network
Agents	Users	Traffic engineer
Control	Offered traffic	Traffic split, feedback
Knowledge	Utility function	Network topology, TE objective
Information received	Feedback	Offered traffic, algorithm type
Action	Use feedback to set offered traffic	Sets feedback and uses offered traffic to set traffic split
Action period	Offered traffic continuously adapted	Traffic split updated at any irregular interval

- Capacity constraint prevents further updates to the traffic split ratio for user 2 (Example 2.3).

We pause here to point out the critical differences between our work in the most general setting and prior literature. Most importantly, we have relaxed the fixed offered assumption in traffic engineering and consider an elastic traffic modeled by (NUM). We stress again that our main perspective lies with the traffic engineer, though our work has several implications for the congestion control framework as well. First, we adopt the view that network infrastructure is owned and controlled by the traffic engineer. Thus, the traffic engineer has a complete view and control of the network topology while users could obtain information about the paths used via feedback but beyond that have limited to no knowledge of the network topology and possible paths to destination. This is in direct contrast with [58] where users could choose which path to use and with [106] where edge routers act independently of each other. Second, prior work with globally optimal convergence results, notably [46, 82], require synchronous updates by the user and traffic engineer. This is a timescale mismatch

as congestion control typically converges to equilibrium in the order of round-trip time while routing is generally updated on a longer timescale. The paper [47] investigates various timescale separation models but its stability and optimality results are limited to the ring topology. Here, we complete the stability and globally optimal convergence proof for the general topology and for the general timescale model where the traffic engineer could update at any arbitrary interval.

2.3 Model Analysis

We first show that the iterative two-state process is stable in general.

Theorem 2.1. *The iterative two-stage process (2.5), (2.6) converges to a fixed point.*

Proof. The sequence

$$\left\{ \sum_i U^i \left(x^i \left(t \right) \right), t = 1, 2, \ldots \right\}$$

converges since it is bounded from above due to capacity constraint, and strictly increasing prior to convergence since at each time $t+1$, $x\left(t\right)$ is a feasible solution of (2.6) and thus

$$\sum_i U^i \left(x^i \left(t+1 \right) \right) \geq \sum_i U^i \left(x^i \left(t \right) \right)$$

If the inequality holds with equality, then the sequence has converged with $\left(Q\left(t+1\right), x\left(t\right) \right)$ being a fixed point of equations (2.5) and (2.6). Note that the convention of always choosing the current solution if optimal prevents the possibility of oscillations. □

Remark 2.1. While we have allowed the traffic engineer to split traffic arbitrarily on all possible paths, the proof still holds even if the traffic engineer is restricted

to picking a fixed number of paths or even just one path for each user. In fact, in the single-path case, the process is guaranteed to converge in finite time since there is only a finite number of single-path routing configurations and the iterative process is guaranteed to pick a different configuration at each iteration. This is in contrast to results from [106] where the single-path case is shown to be unstable. As pointed out earlier, this is because the traffic engineer controls all routers instead of allowing each router to act independently of each other.

The proof shows that the two-stage iterative process improves network utility but the same guarantee does not necessarily hold for traffic engineering objective. Consider the following simple example.

Example 2.2. (No improvement for traffic engineer) Suppose the objective of (TE) is to minimize maximum utilization. The optimal solution for (NUM), however, will always have a bottleneck link, l where $\bar{f}_l = c_l$. This implies that the effort of the traffic engineer is futile as at the end of each iteration, there will always be a link with maximum utilization.

Since the network utility $\sum_i U^i\left(x^i\right)$ strictly increases after every iteration prior to convergence, one could view the iterative process (2.5), (2.6) as trying to solve the joint optimization problem

$$\begin{aligned} \text{(JOINT)} \quad \max_{x \geq 0, Q \in \mathcal{Q}} \quad & \sum_{i \in N} U^i\left(x^i\right) \\ \text{s.t.} \quad & HQx \leq c \end{aligned}$$

First note that (JOINT) is a non-convex problem as HQx is not convex in (Q, x). Equation (2.5) could now be interpreted as an update step that allows one to search $\mathcal{Q}$ for better traffic split ratios. However, the iterative process is not guaranteed to reach the optimal solution of (JOINT) regardless of the cost function Φ as demonstrated in the following example.

Example 2.3. (Capacity constraint prevents update of Q) Consider again the network used for Example 2.1. Initially, the traffic engineer sets 1 unit of offered traffic from user 1 to travel along $A \to D$ and splits the 2 units of traffic from user 2: 1 along $B \to D$ and 1 along $B \to C \to D$. Due to capacity constraints, regardless of the traffic engineering objective V, the optimal solution of (TE) maintains the current traffic split ratios and thus the iterative process has converged. However, the optimal solution of (JOINT) is $2U(1.5) > U(1) + U(2)$ by concavity of U. The iterative process fails to improve the solution further as the current solution is trapped due to capacity constraints.

2.4 Model Refinement

Our goal now is to modify the iterative process such that the incentives of both the users and the traffic engineer are taken into account. Since we have two groups with incentives that are not necessarily aligned, the framework of game theory lends itself naturally. We stress that though we borrow the terminologies and tools of game theory, and we may refer to users as trying to maximize a certain utility, we do not assume users are rational and selfish but instead they act according to the governing algorithm.

If we view the group of users as a single player, then we can view the iterative process (2.5), (2.6) as a two-player game between the player and the traffic engineer. The player and the traffic engineer take turns to make a best response to the opponent's strategy and we have shown in Theorem 2.1 that this best-response dynamics converges to a Nash equilibrium. We define the aggregate surplus or the social value of an offered traffic and traffic split combination

(x, Q) as

$$S(x, Q) = \sum_{i \in N} U^i\left(x^i\right) - \Lambda(HQx)$$

It is then natural to look at the price of anarchy and we show that the price of anarchy depends on the capacity for the two-stage iterative process (2.5), (2.6).

Example 2.4. (PoA depends on capacity) Consider a simple graph with only two nodes. A link with capacity $c > 1$ connect the two nodes and a single user with log utility sends traffic from one node to the other. Suppose the traffic engineering objective is to minimize congestion cost, which we assume is linear in the link flow. In this case, the social value is

$$S(x) = \log(x) - x < 0$$

The socially optimal solution is given by $\bar{x} = 1$ with a corresponding social value of $S\left(\bar{f}\right) = -1$. The social value under (NUM), however, is $S(c) = \log c - c$. Since the social value is always negative, one could interpret the social value as cost instead of negative utility. For cost minimization, the price of anarchy is given by

$$\frac{S(c)}{S\left(\bar{f}\right)} = c - \log c$$

which is arbitrarily large as c increases.

Refining our goal further, we now want to modify the iterative process such that it has a fixed point that is socially optimal. We know from Example 2.3 that a solution may get trapped due to capacity constraints. We also know from Example 2.4 that the price of anarchy is dependent on the capacity. These two observations lead us to consider a relaxation of the capacity constraint. To do that, first we restrict the objective of (TE) to doubly differentiable and additive cost functions, $\sum_l \Phi_l(f_l)$. Next we assume that for all l, Φ_l is large when it is

near or over capacity to penalize capacity overshooting. Thus, capacity constraint will still be reflected if we incorporate the link cost into the objective function. As with the primal formulation (NUM-P), for the users, the capacity constraint is replaced with Φ_l as the barrier function. We thus arrive at the following Gauss-Seidel system [14]:

$$Q(t+1) = \arg\min_{Q\in\mathcal{Q}} \sum_{l\in L} \Phi_l\left(e_l^T HQx(t)\right) \tag{2.8}$$

$$x(t+1) = \arg\max_{x\geq 0} \sum_{i\in N} U^i\left(x^i\right) - \sum_{l\in L} \Phi_l\left(e_l^T HQ(t+1)x\right) \tag{2.9}$$

But we are not quite done with the assumptions yet since a solution may also get trapped at the other extreme where $x^i = 0$.

Example 2.5. (Phantom user) We reuse the network shown in Figure 2.2. There are two users with the same strictly concave utility function $U^i\left(x^i\right) = -\left(x^i\right)^2 + 2x^i$, for $x^i \leq 1$. The first user sends traffic along $A \to D$ and $A \to C \to D$ while the second user sends traffic along $C \to D$. The link-cost function is linear $\Phi_l(f_l) = 3f_l/2$. Initially, user 1 is configured to send all its traffic along the second path, i.e. $q^1_{A\to C\to D} = 1$. With this setup, solving (2.9) gives $x^1 = 0$, $x^2 = 1/4$. On the next iteration, the traffic split ratio remains the same by convention. The Gauss-Seidel system thus converges with a social value of $1/16$. However, the optimal social value is $1/8$ with $\bar{q}^1_{A\to D} = 1$ and $\bar{x}^1 = \bar{x}^2 = 1/4$.

We therefore assume that a solution of (2.9) always gives $\bar{x} > 0$. The assumption makes sense in practice because the traffic engineer would not be able to know the existence of a user if it is not offering any traffic. This assumption is satisfied with, for instance, utility functions U satisfying $\lim_{x\to 0^+} dU/dx = \infty$, e.g. α-fair utility functions [76].

The analogous optimization problem of (JOINT) after the modifications is

the main problem of this paper:

$$\text{(MAIN)} \max_{x \geq 0, Q \in \mathcal{Q}} \quad \sum_{i \in N} U^i \left(x^i \right) - \sum_{l \in L} \Phi_l \left(e_l^T H Q x \right)$$

The problem is not necessarily convex as $\Phi_l \left(e_l^T H Q x \right)$ is not necessarily convex in Q and x. A simple example to show non-convexity is a 1-user setup with a link l satisfying $\Phi_l \left(f_l \right) = f_l$ and $f_l = q_1^1 x^1$. However, we can transform it into a convex optimization problem with a change of variables

$$w_j^i = q_j^i x^i \text{ or } w = Qx \tag{2.10}$$

where w_j^i is now the traffic flow on path j for user i. We obtain

$$\text{(TRANS)} \max_{w \geq 0} \sum_{i \in N} U^i \left(\sum_j w_j^i \right) - \sum_{l \in L} \Phi_l \left(e_l^T H w \right) \tag{2.11}$$

To get back to the original variables, we can use the inverse relationship whenever $x^i > 0$

$$x^i = \sum_j w_j^i, \qquad q_j^i = \frac{w_j^i}{\sum_k w_k^i} \tag{2.12}$$

Note that a solution is optimal for (TRANS) if and only if the corresponding solution after change of variables is optimal for (MAIN).

We now prove the optimality of the Gauss-Seidel system.

Theorem 2.2. *The Gauss-Seidel system (2.8), (2.9) converges to a socially optimal fixed point,* $(\bar{x}, \bar{Q})$.

Proof. The social value $S(x, Q)$ is increasing after each update and is bounded and thus the Gauss-Seidel system converges to a fixed point $(\bar{x}, \bar{Q})$. To show that the fixed point is socially optimal, we rely on Karush-Kuhn-Tucker (KKT) conditions [19]. Both (2.8) and (2.9) are convex optimization problems that satisfy

regularity conditions since the constraints are linear and thus the KKT conditions are necessary and sufficient. The KKT conditions for (2.8) state that $\bar{Q}$ is optimal if and only if there exists Lagrange multiplier λ such that

$$\left(He_j^i\right)^T \Phi'\left(H\bar{Q}x\right) \begin{cases} = & \frac{\lambda^i}{x^i}, \text{ if } \bar{q}_j^i > 0 \\ \geq & \frac{\lambda^i}{x^i}, \text{ if } \bar{q}_j^i = 0 \end{cases} \tag{2.13}$$

where $\Phi'\left(H\bar{Q}x\right) = \left\{\Phi_l'\left(e_l^T H\bar{Q}x\right), l \in L\right\}$. The KKT conditions for (2.9) state that $\bar{x} > 0$ is optimal if and only if

$$\left(U^i\right)'\left(\bar{x}^i\right) = \left(HQe^i\right)^T \Phi'\left(HQ\bar{x}\right) \tag{2.14}$$

To prove optimality, we will show that these KKT conditions together imply the KKT conditions of (TRANS), and thus imply $(\bar{x}, \bar{Q})$ is an optimal solution of (MAIN). First, (TRANS) is convex with linear constraints and thus the KKT conditions are necessary and sufficient. We denote its Lagrangian as $L_S\left(w;\theta\right)$ and set $\bar{w} = \bar{Q}\bar{x}$. For $\bar{w}_j^i > 0$, choose $\theta_j^i = 0$ and apply chain rule and the equations for change of variables and its inverse (2.10), (2.12)

$$\begin{aligned} \frac{\partial L_S\left(\bar{w},\theta\right)}{\partial w_j^i} &= \frac{\partial x^i}{\partial w_j^i}\frac{\partial L_S\left(\bar{Q}\bar{x},\theta\right)}{\partial x^i} + \sum_k \frac{\partial q_k^i}{\partial w_j^i}\frac{\partial L_S\left(\bar{Q}\bar{x},\theta\right)}{\partial q_k^i} && (2.15) \\ &= \left(U^i\right)'\left(\bar{x}^i\right) - \left(HQe^i\right)^T \Phi'\left(H\bar{Q}\bar{x}\right) && (2.16) \\ &\quad + \left(1 - \bar{q}_j^i\right)\left[-\left(He_j^i\right)^T \Phi'\left(H\bar{Q}\bar{x}\right)\right] && (2.17) \\ &\quad - \sum_{k\neq j} \bar{q}_k^i \left[-\left(He_k^i\right)^T \Phi'\left(H\bar{Q}\bar{x}\right)\right] && (2.18) \\ &= 0 && (2.19) \end{aligned}$$

Equation (2.14) implies line (2.16) equals zero, while equation (2.13) implies lines (2.17) and (2.18) cancel off each other. One could easily verify that the other

KKT conditions are satisfied with proper choices of θ and hence $\bar{w}$ is an optimal solution of (TRANS) implying that $(\bar{x}, \bar{Q})$ is the socially optimal solution. □

With Theorem 2.2 in mind, we know we could achieve social optimality if the offered traffic of the users converges to a solution of

$$\max_{x \geq 0} \sum_{i \in N} U^i \left(x^i \right) - \sum_{l \in L} \Phi_l \left(e_l^T H Q x \right) \tag{2.20}$$

So far, our results and examples have not touched on the dynamics of how users vary their offered traffic. We introduce in the modeling section how the expected congestion prices are passed to the users as feedback values and how users react to the feedback values according to either primal or dual algorithm. For the next two sections, we assume all users are running either the primal or the dual algorithm. Without altering the existing algorithms, our goal in these sections is to propose congestion prices such that the offered traffic of the users converges to a solution of (2.20).

2.5 Primal algorithm as a potential game

We know the primal algorithm (2.1), (2.2) is globally asymptotically stable and converges to the solution of (2.20) for the single path case [97]. The crux of the proof is to show that the objective function of (2.20) acts as a Lyapunov function and is, in essence, similar to the proof technique of Theorem 2.1. The proof for the multipath case could easily be generalized using the same Lyapunov function argument or by using the perspective of a diagonally strictly concave game[2] [90]. However, we will take a slightly different path by viewing the proof

[2]One could show that the game that we are considering next is also a diagonally strictly concave game

from the perspective of a potential game [77] because its properties are useful when we relax the timescale assumptions and when we extrapolate the ideas here to the dual algorithm.

Consider the following game where each user i receives utility $U^i\left(x^i\right)$ for an offered traffic of x^i but is charged the sum of link costs of all links used i.e. $\mathbf{1}^T\left(H^i\right)^T\Phi\left(HQx\right)$ where $\Phi\left(HQx\right)=\left\{\Phi_l\left(e_l^THQx\right), l\in L\right\}$. Thus, each user i tries to maximize the total utility:

$$\max_{x^i\geq 0} \quad \Gamma^i\left(x\right)\triangleq U^i\left(x^i\right)-\mathbf{1}^T\left(H^i\right)^T\Phi\left(HQx\right)$$

Lemma 2.3. *For fixed Q, the users are playing an exact potential game with exact potential function*

$$P\left(x\right)=\sum_{i\in N}U^i\left(x^i\right)-\sum_{l\in L}\Phi_l\left(e_l^THQx\right)$$

Proof. We check by definition that, for all $i\in N$

$$\frac{\partial\Gamma^i}{\partial x^i}=\left(U^i\right)'\left(x^i\right)-\left(HQe^i\right)^T\Phi'\left(HQx\right)=\frac{\partial P}{\partial x^i}$$

□

An exact potential game has the property that any user i that unilaterally switches from rate x^i to y^i increases (or decreases) his utility by the same amount as the potential:

$$\Gamma^i\left(y^i,x^{-i}\right)-\Gamma^i\left(x^i,x^{-i}\right)=P\left(y^i,x^{-i}\right)-P\left(x^i,x^{-i}\right) \tag{2.21}$$

This property implies that it is possible to arrive at a Nash equilibrium if the users take turns performing its best response to the current strategy. In addition, it turns out that the Nash equilibrium is an optimal solution to (2.20).

Lemma 2.4. *For fixed Q, the unique Nash equilibrium, x_{NE} satisfies*

$$x_{NE} = \arg\max_{x \geq 0} \sum_{i \in N} U^i \left(x^i \right) - \sum_{l \in L} \Phi_l \left(e_l^T HQx \right) \tag{2.22}$$

Proof. At Nash equilibrium,

$$x_{NE}^i = \arg\max_{x^i \geq 0} U^i \left(x^i \right) - \mathbf{1}^T \left(H^i \right)^T \Phi \left(HQx \right) \tag{2.23}$$

The optimization problem on the RHS of (2.23) is convex, has linear constraints and the KKT conditions state that for each user i

$$\left(U^i \right)' \left(x_{NE}^i \right) \;=\; \left(HQe^i \right)^T \Phi' \left(HQx_{NE} \right)$$

The N conditions together constitute the KKT conditions of (2.22). Uniqueness arises from the fact that (2.20) is strictly concave. □

We now show that the primal algorithm (2.1) generalizes to the multipath case.

Lemma 2.5. *If users update their strategies according to gradient ascent and congestion prices are set to the first derivatives of link cost:*

$$\dot{x} \;=\; \kappa^i \left(x^i \right) \left[\left(U^i \right)' \left(x^i \right) - \left(HQe^i \right)^T \phi \right] \tag{2.24}$$

$$\phi \;=\; \Phi' \left(HQx \right) \tag{2.25}$$

then the strategies converge to an optimal solution of (2.20).

Proof. We show that the potential is always increasing

$$\begin{aligned} \dot{P}\left(x \right) &= \sum_i \frac{\partial P\left(x \right)}{\partial x^i} \dot{x}^i \\ &= \sum_i \kappa^i \left(x^i \right) \left[\left(U^i \right)' \left(x^i \right) - \left(HQe^i \right)^T \Phi' \left(HQx \right) \right]^2 \end{aligned}$$

which is always greater than zero when x is not the Nash equilibrium. □

We have shown two ways for the potential game to converge to the Nash equilibrium, one via best response dynamics and one via gradient ascent. For best response, only one player is modifying the strategy while for gradient ascent, all players are modifying the strategies. From the proofs, it should be clear that these are not the only two ways where the potential game could converge to the Nash equilibrium. We simply require dynamics where the potential is always increasing and where all players get a chance to update their strategies.

Now, we relax the timescale separation assumption and allow the traffic engineer to perform updates at any irregular intervals. We still assume that the update happens instantaneously.

Consider the potential game earlier with N players. We add in the traffic engineer as an additional player who tries to maximize its utility:

$$\max_{Q \in \mathcal{Q}} \Lambda(Q, x) \triangleq \sum_{l \in L} \Phi_l \left(e_l^T H Q x \right)$$

Lemma 2.6. *The game with $N+1$ players is an exact potential game with exact potential function*

$$P(Q, x) = \sum_{i \in N} U^i \left(x^i \right) - \sum_{l \in L} \Phi_l \left(e_l^T H Q x \right)$$

Proof. We check by definition that

$$\frac{\partial V}{\partial q_j^i} = x^i \left(H e_j^i \right)^T \Phi' (HQx) = \frac{\partial P}{\partial q_j^i}$$

□

Theorem 2.7. *Suppose the N users update their strategies according to (2.24) while the traffic engineer sets congestion prices according to (2.25) and performs best response to update Q at any irregular intervals, then the potential game converges to the socially optimal solution.*

Proof. We have shown that the potential is always increasing when the N users update according to gradient ascent. When the traffic engineer performs a best response update to Q, the potential increases instantaneously according to (2.21). Thus the potential is always increasing when the game is not at the Nash equilibrium. The Nash equilibrium is the fixed point of the Gauss-Seidel system (2.8), (2.9) and we have shown that it is socially optimal. □

Remark 2.2. In fact, the traffic engineer only needs to find a new traffic split matrix Q that strictly increases $\Lambda(Q, x)$.

2.6 Modifying dual algorithm

The dual algorithm shown in equations (2.3) and (2.4) have a capacity-dependent congestion price. Since we have relaxed the capacity constraint, we have to modify the price function accordingly. A direct attempt by taking the dual of the primal formulation does not work out but instead the modification will arise naturally from a game theory perspective by investigating the symmetry of equations (2.1), (2.2), and (2.3) of the primal and dual algorithms.

In the primal formulation, users are treated as players while the links are non-player entities providing the congestion price for link usage. The roles and dynamics are swapped for the dual formulation. Users are now non-player entities providing the offering traffic according to

$$x^i = \left(U^i\right)'^{-1}\left(\left(HQe^i\right)^T\phi\right) \tag{2.26}$$

Note that this is the best response for the users in the potential game. Each link

l is now a player maximizing the utility

$$\max_{\phi_l \geq 0} \beta_l(\phi_l) \triangleq -\frac{1}{2}\left(\Phi_l' - \phi_l\right)^2$$

Note that the links are doing best response in the potential game.

We are now ready to prove an analogous result of Lemma 2.5 and Theorem 2.7 for the dual algorithm.

Lemma 2.8. *If users set offered traffic according to (2.26), and links update their strategies according to gradient ascent:*

$$\dot{\phi}_l = \Phi_l' - \phi_l \tag{2.27}$$

then the strategies converge to an optimal solution of (2.20).

Proof. From (2.26), we obtain the time derivative of

$$\left(U^i\right)''\left(x^i\right)\dot{x}^i = \left(HQe^i\right)^T\dot{\phi} \tag{2.28}$$

Using (2.26), (2.27) and (2.28), one can show that the potential P is always increasing as

$$\dot{P}(x) = \sum_i -\frac{\left[\left(U^i\right)'\left(x^i\right) - \left(HQe^i\right)^T\Phi'\right]^2}{\left(U^i\right)''\left(x^i\right)} \quad \geq 0$$

□

Theorem 2.9. *Suppose the N users update the offered traffic according to (2.26), and the traffic engineer sets the L links to update according to (2.27) and performs best response update to Q at any irregular intervals, then the process converges to the socially optimal solution.*

Proof. Analogous to Theorem 2.7. □

We have assumed that all the users adopt the same algorithm in the last two sections. Our final contribution is to consider the case when there is a mix of users running primal and dual algorithms.

2.7 Heterogeneous users and feedback

The result in this section is a straightforward combination of the results from the prior two sections. Suppose the users are divided into two non-intersecting subsets: a set N^p of users running primal algorithm and a set N^D of users running dual algorithm. The traffic engineer is assumed to know the type of algorithm each user is running. A different congestion price and thus feedback is calculated for each algorithm. For the primal algorithm, the congestion prices ϕ^p update according to (2.25) while for the dual, ϕ^D update according to (2.27).

Theorem 2.10. *If users update their offered traffic and links update the offered traffic and congestion prices according to*

$$\begin{aligned}
\dot{x}^i &= \kappa^i\left(x^i\right)\left[\left(U^i\right)'\left(x^i\right) - \left(HQe^i\right)^T\phi^P\right],\ i \in N^P \\
x^i &= \left(U^i\right)'^{-1}\left(\left(HQe^i\right)^T\phi^D\right),\ i \in N^D \\
\phi_l^P &= \Phi_l',\ \forall l \in L \\
\dot{\phi}_l^D &= \Phi_l' - \phi_l^D,\ \forall l \in L
\end{aligned}$$

while the traffic engineer performs best response update to Q at any irregular intervals, then the process converges to the socially optimal solution.

Proof. We show that the potential P is always increasing

$$\dot{P}\left(x\right) = \sum_{i \in N^p} \frac{\partial P\left(x\right)}{\partial x^i}\dot{x}^i + \sum_{i \in N^D} \frac{\partial P\left(x\right)}{\partial x^i}\dot{x}^i$$

As shown in the proofs of Lemma 2.5 and 2.8, the sum of the two terms are always positive except at the optimal solution of (2.20). The rest of the proof is analogous to Theorem 2.7. □

CHAPTER 3

TRAFFIC DYNAMICS OF PACKET-SWITCHED NETWORK (MILLISECONDS - SECONDS)

3.1 Introduction

In the previous chapter, we have investigated how users and network operators interact over a timescale of minutes to hour. As users update at a timescale of one second or less, by the time a network operator comes around to performing traffic engineering, the users would have updated sufficient number of times such that the network operator could essentially treat the current network condition as being stable. Such an assumption forms the basis of the Gauss-Seidel system (2.8), (2.9) from the previous chapter. We have also assumed that the traffic engineering step is instantaneous when in fact it is not as it takes time for any update to propagate through the network. Such an update propagation is in the same order as round-trip time and as such a more thorough investigation requires us to look at a finer timescale where routing update and user update occur in the same timescale and that is the focus of this chapter.

We first lay down the framework of network traffic dynamics. The framework is general, as user requirements would not be limited to just throughput and network operators would not necessarily be required to do source routing as in the previous chapter. Another key difference here is that we are essentially looking at a dynamic system with users and operators updating at the same time and as such keeping track of time is important. As such, timing information that is previously neglected such as queuing delay, propagation delay, etc. are now included for all the network elements.

We next specialize the framework to investigate how network operators could deal with the consistent and swiftness challenges during traffic engineering. The consistency property that we are interested in is congestion-free, i.e. the link capacity is not exceeded at all times during the routing reconfiguration. The network operator starts with an initial routing configuration and has determined the targeted optimal routing configuration. Before an update has fully propagated through the network, some of the links would contain a mix of un-updated traffic flows following the old configuration and updated traffic flows following the new configurations. Congestion could occur due to this mix of traffic and the objective is to avoid this while completing the reconfiguration in the least amount of time possible.

3.2 Framework of Network Traffic Dynamics

Though parts of the framework here overlap with the model in the previous chapter, due to the generality of the framework, the notations and conventions of this chapter differ from the previous chapter. To prevent confusion, we reintroduce the overlapping part and the associated notations.

We are given a network with a set V of switches and a set L of directed links. The network, which is controlled by a network operator, has a set of N users.

Each user i represents a particular source-destination pair (s_i, d_i) (SD pair), $s_i, d_i \in V$. Each user controls the amounts of traffic flowing into the network either by directly setting the offered traffic $x_i(t)$ or by altering the rate of change of the offered flow, $\dot{x}(t)$. In addition, the user's decision on its traffic flow could be dependent on the feedback, z_i that it received. We will go into the details of

Table 3.1: How users could control the input traffic

	Open-loop	Closed-loop
Direct	$x(t)$	$x(t, z)$
Rate of change	$\dot{x}(t)$	$\dot{x}(t, z)$

the feedback shortly but for now, the possible methods of control by a user are listed in Table 3.1. The different methods of control models different scenarios. For instance, a direct control without feedback could model open-loop system such as a datacenter network where the incoming traffic over a daily cycle could be reasonably estimated. In contrast, controlling with feedback could model closed-loop system such as congestion control as in the previous chapter. The users could either be homogeneous, in which case all the users adopt the same method of control, or the users could be heterogeneous, in which case, there are different classes of users adopting different methods of control.

Each user is associated with a set of acyclic paths P_i starting at source switch s_i and ending at destination switch d_i. Each acyclic path p is an ordered tuple $(s_i, l_1, v_1, \ldots, l_m, d_i)$ of switches and links. For each link l along an acyclic path $p \in P_i$, q denotes the subpath from s_i to the switch prior to link l. We denote the one-to-one correspondence between a path and its subpath to link l as $q \underset{l}{\leftrightarrow} p$. We denote the collection of all such subpaths from any source switch to link l as Q_l and from a particular source s_i as Q_{il}. We will sometimes abuse notation and write $p \in Q_l$ to mean $\left\{ p : p \underset{l}{\leftrightarrow} q \in Q_l \right\}$. Note that it is possible for Q_l to have duplicated elements if there are multiple paths with the same subpath till link l.

As we have seen in the previous chapter, for closed-loop systems, a user's control over its input traffic could be influenced by underlying requirement constraints or objective function. The objective could be a function of throughput, average end-to-end delay or a weighted sum of both. Similarly, the constraints

could be hard constraints requiring a minimum amount of throughput or a guaranteed lower bound on end-to-end delay or soft constraints that penalizes the objective function if the requirements are not satisfied. Each possible user control could then be interpreted as improving the objective or getting closer to satisfying the requirements; or the reverse. We have seen examples of this in the previous chapter: gradient ascent/descent wherein the input traffic is changed at a rate that improves its objective, and best-response wherein the input is set to the optimizing rate given the feedback. By exploiting the convexity or monotonicity of the constraints or the objective, such system will converge to a solution that satisfies the constraints or optimizes the objective.

The network operator controls how the user traffic is routed in the network. Depending on the underlying routing protocol governing the network, the network operator could have varying degrees of control over routing:

- $\alpha_{ip}(t)$: For each user i, the network operator tells the source router the ratio of traffic to send along each acyclic path p from s_i to d_i. This assumes the network is setup with tunnels overlay for each of the path and each of the intermediate router simply forwards traffic according to the tunnels.
- $\alpha_{vsd}(t)$: For each router v, the network operator specifies the split ratio to send along each of its outgoing link for traffic from router s heading for router d.
- $\alpha_{vd}(t)$: For each router v, the network operator specifies the split ratio to send along each of its outgoing link for traffic heading for switches d.

The degree of control is a reflection of the available information in descending order: routing according to path, according to source-destination or according

to destination. The information used also reflects on the size of the forwarding table of each router: more information requires larger forwarding tables. The number of paths within a network is exponential in L and V; the number of source-destination pair is polynomial in V; and the number of destination switches is linear in V. As such, network sizes and practical limits on the forwarding tables would either determine the possible control, or would require additional constraints such as source routing over a limited number of predetermined paths instead of over all paths. Similar to users, network operators could have an underlying objective or constraint governing its control.

The network elements are associated with various delays. Each link l is associated with a propagation delay t_l, and each switch v is associated with a processing delay t_v, set of incoming links L_v^+, and a set of outgoing links L_v^-. For each of the outgoing link $l \in L_v^-$, there is a buffer that builds up at a rate of

$$\dot{B}_l(t) = \left(\frac{f_l(t) - c_l}{c_l}\right)_{B_l(t)}^+ = \begin{cases} \frac{f_l(t)-c_l}{c_l} & , B_l(t) > 0 \\ \max\left(0, \frac{f_l(t)-c_l}{c_l}\right) & , B_l(t) = 0 \end{cases}$$

where c_l is the capacity of link l and $f_l(t)$ is the total incoming flow for link l. Assuming work conservation, the total outgoing flow for link l is given by

$$g_l(t) = f_l(t) - (f_l(t) - c_l)_{B_l(t)}^+ = \begin{cases} c_l & , B_l(t) > 0 \\ \min(f_l(t), c_l) & , B_l(t) = 0 \end{cases}$$

Under a first-come-first-served queue discipline, the queuing delay would simply be the size of the buffer, i.e. $B_l(t)$.

Depending on the control over routing that the network operator has, the incoming and outgoing flows for a link could be further broken down. If the network operator routes according to path, then we can break down into sub-

flows over each path using link l:

$$\begin{aligned}
\sum_{p \in Q_l} g_{lp}(t) &= g_l(t) \\
g_{kp}(t - t_k - t_v) &= f_{lp}(t),\ k \in L_v^+, l \in L_v^- \\
\sum_{p \in Q_l} f_{lp}(t) &= f_l(t)
\end{aligned}$$

Relating the incoming sub-flows $f_{lp}(t)$ with the outgoing sub-flows $g_{lp}(t)$ would require knowing the queuing discipline, and the scheduling discipline for how the capacity is allocated when the incoming flow exceeds capacity. For instance, if at time $t = 0$, the buffer is empty, $B_l(0) = 0$, the incoming flow exceeds capacity, $f_l(0) > c_l$, and the capacity is shared equally (round-robin), then

$$g_{lp}(0) = \frac{c_l}{\left|\{p : f_{lp}(0) > 0\}\right|}$$

The sub-flows can be broken down similarly for when the network operator routes according to source-destination or destination only.

Finally, we have the feedback. The feedback is usually a compilation of the network condition over a path. For instance, the feedback z could be the available bandwidth over path p:

$$z = \min_{l \in p}(c_l - f_l)$$

where f_l is measured at the time when the flow passes through link l. The feedback could be dependent on the buffer size, the queuing delay, the incoming or outgoing flow, the link capacity, on the link cost which is part of the operator's objective function. The network operator could choose which of these to send back as feedback or to set the feedback directly.

To summarize, to characterize the network fully, one would need to specify the users, the network operator and the network elements. The users controls the **input rate of traffic demand**, which could be dependent on feedback

(**closed-loop**) or not (**open-loop**). For a closed-loop system, the users could have an associated **objective** function driving the control. The users could all be running the same type of control (**homogeneous**) or different types of control (**heterogeneous**). The network operator controls the **routing**, which could be **path**-based, **source-destination**-based, or **destination**-based. The operator could similarly have an **objective** function driving the routing. The network elements have associated **delay** values. Each link has a buffer with an associated **queue** and **scheduling discipline**. **Feedback** is usually a compilation of the network condition along path.

3.3 Application: Fast Congestion-Free Routing Reconfiguration

We specialize the framework of network traffic dynamics to investigate how network operators could deal with the consistent and swiftness challenges during traffic engineering. The consistency property that we are interested in is congestion-free, i.e. the link capacity is not exceeded at all times during the routing reconfiguration. The network operator starts with an initial routing configuration and has determined the targeted optimal routing configuration. The objective here is to come up with a sequence of routing update steps taking as short amount of time as possible to reconfigure to the targeted configuration.

3.3.1 Background

The study of avoiding undesirable transient behavior during network changes could be traced back to the distributed routing protocols. The most well-known

example is the routing loop arising from the count-to-infinity problem [13] due to link or node failure in a network running distance vector routing protocols such as RIP [49, 71]. Workarounds such as split horizon and poison reverse [15] could remove some but not all cases of the count-to-infinity problem. RIP is one of the oldest routing protocols though, and it determines the shortest path by hop, i.e. each link has a static weight of 1.

The predominant distributed routing protocols of today are link state routing protocols represented by OSPF [79] and IS-IS [22]. For each link, network operators assign the weights to be used in the shortest path computation of these protocols. As such, in contrast to RIP, network operators could indirectly influence the network routing to achieve their objectives via proper link weight assignment. The earliest work establishes the hardness of the link weight assignment problem [37]. From there, we have two different but related lines of research on link weight assignment. The first is about *dealing* with transient behavior such as traffic change or link failures [38, 81]. The primary concern here is to compute link weights that are robust to traffic demand changes and periodic link failures. The other, which we are extending from, is about *avoiding or minimizing* undesirable transient behavior such as loops [41], disruptions [87] or service outages [102] during reconfigurations. The primary concern is to find an ordering to perform the reconfiguration to avoid undesirable behavior. These are usually combinatorial optimization problems [80] and are as such, NP-hard to solve.

Recent development with a centrally controlled network as enabled by the OpenFlow framework [6] allows network operator to sidestep the link weight assignment problem by controlling the network routing directly. The direct con-

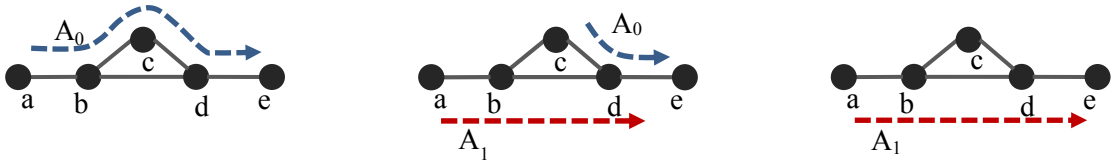
trol reduces the problem complexity and allows for a more in-depth investigation. In particular, [51] gives a linear programming formulation for achieving congestion-free routing reconfiguration with the minimal number of steps. Here, we look at the congestion-free routing reconfiguration problem via the framework of network traffic dynamics. We introduce the transient congestion the timing uncertainty phenomena in the next two sections before comparing our formulation against [51].

3.3.2 Transient Congestion

Before an update has fully propagated through the network, the network would contain a mix of un-updated traffic flows following the old configuration and updated traffic flows following the new configurations. We illustrate next how congestion could occur during reconfiguration due to propagation delay differences between the updated and un-updated traffic flows.

Example 3.1. (Propagation delay) Consider Figure 3.1 in which we have a 1-user network performing a one-shot update (i.e. a one-step update) going from the initial configuration in Figure 3.1a to the final configuration in Figure 3.1c. Flow A_0 takes a longer time compared to flow A_1 to travel from b to d due to differences in propagation delay. As such, assuming flow continuity, i.e. the network continues to carry the same amount of traffic in between the update, Figure 3.1b shows link (d, e) containing a mix of old and new traffic flows. Congestion occurs if their total rate exceeds the link capacity.

There are two ways to avoid such congestion. First, instead of moving flow A_0 to A_1 in one-shot, we could stagger it into multiple updates where only a



(a) Initial configuration (b) Transient behavior (c) Final configuration

Figure 3.1: Congestion due to reconfiguration

fraction of the flow is moved each time. Second, if we do not care about the network carrying a constant traffic demand at all times, then we could do a one-shot update where we only send flows through A_1 after waiting sufficient time to account for the differences in propagation delay. The first method is less disruptive though slower and sometimes, the method is infeasible. For instance, when the carried flow of A_0 is equal to the capacity of link (d, e), then there is no way to reconfigure without causing transient congestion. In such a situation, we may have to trade off some traffic demand. We will examine both methods in this chapter.

3.3.3 Timing Uncertainty

For flows from different sources, we have an additional factor that could cause congestion. When the network operator sends instructions to update the source switches, these instructions might not be executed in a synchronous fashion. There are three timing-related reasons. First, there is no guarantee that the instructions to update would arrive at each source switch at the same time due to difference in distances from the controller. Second, even if the updates arrive synchronously, each switch may still take a variable amount of time to properly execute depending on the load and state of the switches. Finally, instruc-

tions with an added condition on the execution time would not work either as the clock of source switches are not perfectly synchronized. Additional timing uncertainty arises when packets transition through a router, as mentioned in Chapter 2.

The timing uncertainties mean that sometimes we will not able to tell exactly when an old flow would clear a link or a new flow would arrive. Instead, we now have a range of time for each flow. When the ranges of different flows overlap, we no longer have just one specific order of events but instead multiple possible mix of old and new flows could occur and we need to take all of them into account. Sometimes, when the timing uncertainty is large, all possible combinations of old and new flows are possible and we call this worst-case scenario as *order-oblivious*.

3.3.4 Benefits of Timing Information

Timing information means that we no longer have to consider the order-oblivious worst-case scenario as in [51]. To illustrate, consider again Figure 3.1 but in reverse: Figure 3.1c is the initial configuration while Figure 3.1a is the final configuration. Due to propagation delay differences, flow A_1 will clear link (d, e) before flow A_0 arrives and thus a one-shot update is congestion-free. Without timing information, an order-oblivious case would consider a one-shot update as causing congestion if the total rate of A_0 and A_1 exceeds capacity of (d, e). In this case, there is either no feasible update or a feasible update would take more than one step to complete.

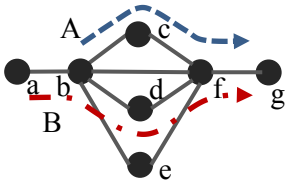
Timing information allows us to know how long it would take an update

to propagate through the entire network, which in turns let us know when we could start the next update. To be precise, having an update fully propagating through the network means that none of the flows following the old configuration is in the network and all the flows following the new configuration has arrived at the destination switches. Knowing this means that we no longer have to wait an arbitrarily long period of time to guarantee that the update has propagated through the entire network. In addition, since we know the time required for each update, we could optimize the update sequence for the total time required, instead of number of update steps, which is not necessarily optimal.

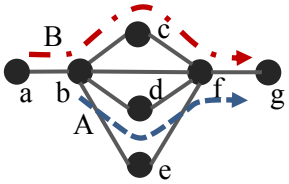
Example 3.2. (Shorter update is not necessarily faster) Consider Figure 3.2. Flow A and B both carry a traffic demand of 1. Link(f, g) has a capacity of 3, link(b, f) has a capacity of 0.5, and all others have a capacity of 1. All links have a propagation delay of 10 ms except for link (e, f) which has a propagation delay of 100 ms. For this example, we only consider propagation delay and ignore other timing effects such as processing delay at switches and the aforementioned timing uncertainty.

The minimum number of congestion-free update steps to achieve the target configuration is two. Flow A moves its flow to the spare path (b, e, f, g) and flow B move its flow to (a, b, c, f, g) in the first time step. In the second time step, flow A move its flow from path (b, e, f, g) to the target configuration. The times for the update to propagate through the network for each time step are 120 ms and 120 ms, respectively for a total of 240 ms. However, the time-optimal update steps require three steps but take less time. First, flow A move half of its flow to spare path (b, f, g) and flow B move half of its flow to (a, b, c, f, g). In the second step, flow A move another flow of 0.5 from path $(b, c, f, g)$to path (b, d, f, g) and flow B move the rest of the flow to the target configuration. In the third step,

flow A move the other half of the flows to the target configuration. The times for each time step are 40 ms, 40 ms, and 30 ms, respectively for a total of 110 ms.



(a) Initial flow configuration



(b) Target flow configuration

Figure 3.2: Minimizing number of update steps does not necessarily minimize update time

3.4 Model Formulation

The congestion-free requirement means that there will be no buffer build-up and as a consequence, no queuing delay. We only need to add in the timing uncertainties to complete the picture. For each switch v, the traffic flow will encounter a minimum switch transition delay, t_v plus an uncertainty δ_v. We associate with each source switch s_i, a timing uncertainty variable δ_i representing the maximum asynchronicity relative to other source switches. The notations used here are summarized in Table 3.2. With these information, for each link l, we could now compute the minimum and maximum time for when the newly updated traffic would be in place for subpath q to respectively be

$$\begin{aligned} w_q^{min} &= \sum_{l \in q} t_l + \sum_{v \in q} t_v \\ w_q^{max} &= w_q^{min} + \delta_i + \sum_{v \in q} \delta_v \end{aligned}$$

Currently in place in the network is a traffic demand x_i for each user and an initial routing configuration $x_{p_i}(0)$ for each path $p_i \in P_i$ of each user. We

Table 3.2: Commonly used variables.

For each link $l \in L$ (index on subscript)	
c_l	Capacity on link $l \in L$
f_l	Total flow on link l
For each SD pair $i \in N$ (index on superscript)	
x_i	Offered flow into the network
$s_i, d_i \in V$	Source and destination switch
P_i	Set of acyclic paths from origin to destination
Q_l	Set of subpath from source to link l
t_l	Propagation delay
For switch $v, w \in V$ (index on subscript)	
t_v	Transition delay
δ_v	Transition timing uncertainty
For each path p_i with source switch s_i	
x_{p_i}	Flow on path $p_i \in P_i$
δ_i	Update timing uncertainty
$\left[w_q^{min}, w_q^{max}\right]$	Time range for when traffic following new updates rules would arrive at link l on subpath $q \in Q_l$

are given a target routing configuration y_{p_i}. The two routing configurations are assumed to be congestion-free, i.e. the traffic flow on each link, $f_l = \sum_{p \in Q_l} x_p$ does not exceed capacity. Note that we are changing routing configuration due to changing network conditions. The network as specified by V, L and x_i are the network after changes such as link failure has occurred. Our objective is to find a sequence of update steps to reconfigure the network from the initial state to the target state, while remaining congestion-free during the update steps. We impose the assumption that the network carries the target traffic demand during each update step. We also assume that we do not have to worry about out-of-order packets, as they are taken care of by the higher layers.

3.5 Congestion-free Constraint

We now figure out what constraints are required to capture the congestion-free requirement during the update steps. We start by illustrating with a simple example.

Example 3.3. (Congestion-free constraints) Consider a link l shared by two paths $p_i \underset{l}{\leftrightarrow} q_i$ and $p_j \underset{l}{\leftrightarrow} q_j$ of users i and j. Once updated, the new traffic flow from user i will arrive at link l within the time range $\left[w_{q_i}^{min}, w_{q_i}^{max}\right]$ while for user j, it is $\left[w_{q_j}^{min}, w_{q_j}^{max}\right]$. When the time ranges do not overlap, i.e. $w_{q_i}^{max} < w_{q_j}^{min}$ (or $w_{q_j}^{max} < w_{q_i}^{min}$), the new flow from user i will always arrive earlier than from user j, this translates to the constraint:

$$\begin{aligned} x_{p_i}(a+1) + x_{p_j}(a) &\leq c_l \\ x_{p_i}(a+1) + x_{p_j}(a+1) &\leq c_l \end{aligned}$$

When the time ranges overlap, say $w_{q_i}^{max} \geq w_q^{min} \geq w_{q_i}^{min}$, either flow could arrive before the other. In this case, the timing information does not provide any additional information and we are in the order-oblivious case. The constraints could be written simply as

$$\max\left(x_{p_i}(t), x_{p_i}(t+1)\right) + \max\left(x_{p_j}(t), x_{p_j}(t+1)\right) \leq c_l$$

The example shows the potential for a reduced constraint set compared to the order-oblivious case using timing information. In general, one could construct the constraint set for each link either by construction or by elimination. One could try to step through the time axis, note the overlap or non-overlap in the time ranges and construct all the possible mix of old and new flows as suggested by Example 3.3. Alternatively, one could start with the full $2^{|Q_l|}$ linear

constraints from the order-oblivious case and check if each of them is possible to occur. We will take the elimination approach as it is heuristically simpler.

For a link l, each of the $2^{|Q_l|}$ constraint can be expressed in the following form:

$$\sum_{p_i \in Q_l^-} x_{p_i}(a) + \sum_{p_i \in Q_l^+} x_{p_i}(a+1) \leq c_l \tag{3.1}$$

where Q_l^- and Q_l^+corresponds to the set of traffic flow where the old update has not cleared link l and the new update has propagated to link l, respectively. They are non-overlapping partitions of Q_l, i.e. $Q_l^- \cap Q_l^+ = \emptyset$, $Q_l^- \cup Q_l^+ = Q_l$. Due to space consideration, we denote the left-hand side of (3.1) as $f_l^a\left(Q_l^+\right)$. Each partition implies a certain ordering with respect to flow arrivals and the question here is that: could this scenario happen? If it does, we collect the partition into a constraint set $\mathcal{Q}_l^a$; if not, it is a constraint that we do not need to care about. Reusing example 3.3, suppose $w_{q_i}^{max} < w_{q_j}^{min}$ but the partition $Q_l^- = i$ and $Q_l^+ = j$ implies a constraint that could never happen, and thus could be ignored. Each element $Q_l^+ \in \mathcal{Q}_l^a$ is a partition that satisfies two conditions:

(i) For each user i, the realized timing uncertainty at the switch is the same for all its traffic flows so we could remove it from consideration. Each of the traffic flows in Q_{il}^+ have its update propagated to link l and thus at least a time of $\max_{q \in Q_{il}^+} w_q^{min}$ have passed. Similarly, each of the traffic flows in Q_{il}^- have its update not propagated to link l yet and thus at most a time of $\min_{q \in Q_{il}^-} w_q^{max} - \delta_i$ have passed. For the scenario to be possible implies that

$$\max_{q \in Q_{il}^+} w_q^{min} < \min_{q \in Q_{il}^-} w_q^{max} - \delta_i \tag{3.2}$$

(ii) Across different users, timing uncertainty matters. Following the same reasoning as in (i), each of the traffic flows in Q_l^+ have its update propagated

to link l and thus at least a time of $\max_{q \in Q_{il}^{+}} w_q^{min}$ have passed. While each of the traffic flows in Q_{il}^{-} have its update not propagated to link l yet and thus at most a time of $\min_{q \in Q_{il}^{-}} w_q^{max}$ have passed. For the scenario to be possible implies that

$$\max_{q \in Q_l^{+}} w_q^{min} < \min_{q \in Q_l^{-}} w_q^{max} \tag{3.3}$$

3.5.1 Robustness

In practice, it could be difficult to estimate all the delays and timing uncertainties accurately. What we could do is to find bounds for the propagation delay in the form of $t_l^{min} \leq t_l \leq t_l^{max}$ and similarly for switch transition delay in the form of $t_v^{min} \leq t_v \leq t_v^{max}$. While for the timing uncertainties, we find upper bounds $\delta_v^{max} \geq \delta_v$ and $\delta_i^{max} \geq \delta_i$. The minimum and maximum time for a newly updated traffic to travel a subpath q is now

$$\bar{w}_q^{min} = \sum_{l \in q} t_l^{min} + \sum_{v \in q} t_v^{min} \tag{3.4}$$

$$\bar{w}_q^{max} = \sum_{l \in q} t_l^{max} + \sum_{v \in q} t_v^{max} + \delta_i^{max} + \sum_{v \in q} \delta_v^{max} \tag{3.5}$$

We now step through the same steps to construct the constraint set by checking (3.2) and (3.3). Since $\bar{w}_q^{min} \geq w_q^{min}$ and $\bar{w}_q^{max} \geq w_q^{max}$, a partition Q_l^{+} could satisfy the constraints even though the same constraints would have been violated if the actual values are used instead. This implies that we have a stricter constraint set, and any solution obtained with these bounds are actually feasible though suboptimal. In other words, how accurately we could determine the delays and uncertainties would affect how close to optimal the solution is. In the worst case when the upper bounds are large, we degenerate to the order-oblivious scenario.

3.6 Optimization Problem With Constant Demand

Our objective is to come up with a sequence of updates to get from the initial to the final configuration in the minimum amount of time where the traffic demand is constant throughout the process. The last piece of information that we need is the amount of time it takes for an update to be completed before we can take the next. Without any timing information, the naive approach is to wait for a sufficiently long period of time to guarantee all updates have propagated throughout the network. Here, we can define precisely the waiting time required to be

$$u_a = \max\left\{w_{p_i}^{max} : x_{p_i}(a+1) \neq x_{p_i}(a)\right\} \tag{3.6}$$

By convention, we take the expression to be zero if none of the traffic flows are updated.

The optimization problem could be formulated as: given the initial and final configuration $x_i, x_{p_i}(0)$ and y_{p_i}

$$\min \sum_{a=0}^{b} u_a \qquad \text{(OPT)} \tag{3.7}$$

$$\text{s.t. } f_l^a\left(Q_l^+\right) \leq c_l, \qquad Q_l^+ \in \mathcal{Q}_l^a, \forall l \in L, 0 \leq a \leq b \tag{3.8}$$

$$\sum_{p \in P_i} x_p(a) = x_i, \qquad \forall i \in N, 1 \leq a \leq b \tag{3.9}$$

$$x_{p_i}(a) \geq 0, \qquad \forall p_i \in P_i, \forall i \in N, 1 \leq a \leq b \tag{3.10}$$

$$u_a \geq w_{p_i}^{max} z_{p_i}(a), \forall p_i \in P_i, \qquad \forall i \in N, 0 \leq a \leq b \tag{3.11}$$

$$\left|x_{p_i}(a+1) - x_{p_i}(a)\right| \leq \alpha_i \cdot z_{p_i}(a), \quad \forall p_i \in P_i, \forall i \in N, 0 \leq a \leq b \tag{3.12}$$

$$z_{p_i}(a) \in \{0,1\}, \qquad \forall p_i \in P_i, \forall i \in N, 0 \leq a \leq b \tag{3.13}$$

where $x_{p_i}(b+1) = y_{p_i}$ and $\alpha_i = \min\{x_i, c_{p_i}\}$ where $c_{p_i} = \min\{c_l : l \in p_i\}$ is the bottleneck link of path p_i. The variables in this optimization problem are b, the number of update steps-1, u_a, the update time of step a, $x_{p_i}(a)$, the route configuration during time step a, and $z_{p_i}(a)$, the binary variables representing the changes to routes. The constraints (3.8) - (3.10) are the feasibility constraints, as if we can find b and $x_{p_i}(a)$ satisfying these constraints, then we have a feasible update steps. Constraints (3.11) - (3.13) are required for an equivalent representation of the objective function (3.6).

The optimization problem (OPT) is not necessarily feasible. For instance, using example 3.1, suppose flow A_0 carries a demand of 1 and link (d, e) has a capacity of 1. Then no update sequence exists to get from the initial to the final configuration. The problem here is that the links are fully occupied and there is no room for the traffic flow to move around. We will discuss what could be done when the problem is infeasible in Section 3.7.

Note that the feasibility constraints (3.8) - (3.10) are the feasibility constraints of [51] with the order-oblivious constraint replaced with a looser constraint (3.8) that takes into account timing information. As such, borrowing the idea from [51], a feasible solution is guaranteed to exist if all links have at least a fraction $1/k$ of spare capacity at the initial configuration. In this case, we have a feasible monotonic solution with $\lceil k \rceil - 1$ steps. This feasible solution has a total update time of say T, and we can use this to upper bound the optimal number of update steps as T/u_{min}, where $u_{min} = \min_{p_i}\left\{w_{p_i}^{max}\right\}$ is the minimal time for an update to happen. Fixing$b$ to the upper bound turns (OPT) into a mixed integer linear program (MILP) [80], which can be solved exactly using methods such as branch and cut or approximately using linear programming relaxation [1]. Solving the

[1]LP relaxation is obtained by replacing constraint (3.13) with $z_{p_i}(a) \geq 0$. Solving the LP

MILP exactly gives an optimal solution $x^*_{p_i}(a)$ where for a certain b^*, $x_{p_i}(a) = x_{p_i}(a)$ for all $a \geq b^*$. We can truncate these non-updates and together b^* and $\left\{x_{p_i}(a) : 1 \leq a \leq b^*\right\}$ are the optimal solution to (OPT).

3.7 Dealing with heavy traffic

When the network is heavily congested, the optimization problem (OPT) might be either infeasible or would require a long update time. In the former case, we no longer have a feasible update sequence that guarantees the network can continue to carry the desired amount of traffic at all times. In the latter, the network can continue to carry the desired traffic though by the time the target configuration is reached, the network conditions could have long since changed and it is no longer optimal. In these cases, it could be desirable to trade off some demand in exchange for a fast update. We propose here a two-step congestion-free update solution. The basic idea is that we set the traffic flows to follow the target configuration in the first update step. Since the capacity constraint is violated on some links, we reduce the traffic demand on some of the flows. At the same time, we would like to minimize the demand reduction and hence we could formulate the following linear program:

$$\min \quad \sum_{i \in N} \left(y_i - \sum_{p \in P_i} x_p(1) \right) \tag{3.14}$$

$$\text{s.t.} \quad f^0_l\left(Q^+_l\right) \leq c_l, \qquad Q^+_l \in \mathcal{Q}^a_l, \forall l \in L \tag{3.15}$$

$$\sum_{p \in P_i} x_p(1) \leq x_i, \qquad \forall i \in N \tag{3.16}$$

$$x_{p_i}(1) \geq 0, \qquad \forall p_i \in P_i, \forall i \in N \tag{3.17}$$

$$y_{p_i} \geq x_{p_i}(1), \qquad \forall p_i \in P_i, \forall i \in N \tag{3.18}$$

relaxation gives a feasible albeit suboptimal solution without requiring any rounding.

At the second step, we simply increase the reduced demand to the target configuration. Due to the reduced demand, the update step is guaranteed to be congestion-free. For any given network, the two-step congestion-free update solution has an update time that is upper bounded by $2 \cdot \max\left\{w_p^{max} : p \in P_i, i \in N\right\}$. As such, it could be preferred not just in the heavy-traffic case, but also when a guaranteed update time is desired for easier network planning.

3.8 Improving update time

Currently, our model requires us to wait for all updates to propagate to the destination switches before we can start the next update. In general though, we frequently can start the next update before the previous update propagates to all destination switches.

Example 3.4. (Shorter update time) We reuse example 3.2 and consider the three-step optimal update sequence. The end of flow B from step 0 reaches switch b 10 ms after the first update. Thus we could now perform the second update. After another 10 ms, the end of flow B from step 1 reaches switch b and we can now perform the third update. The third update will take 30 ms for the old flows to clear the network. The total update time is 50 ms, saving us 60 ms.

The example shows that we could perform the third update even before the first update has fully propagated through the network. The problem gets increasingly more complicated for each additional previous step that we need to consider. Here, we show how one could improve the update time by only considering one previous update and assuming all other updates prior to this has fully propagated through the network. Suppose the current update step is a,

we need to figure out how long we need to wait before we can perform update step $a+1$. The waiting time needs to be long enough such that at any link l, we do not have flows from step $a-1$, a and $a+1$ coexisting. Consider each link l individually, the old flows would take at most

$$\Delta_l^- = \max\left\{ w_q^{max} : x_{p_i}(a-1) \neq x_{p_i}(a), p_i \underset{l}{\leftrightarrow} q, q \in Q_l \right\}$$

to clear the link. Additionally, the fastest new flows of step $a+1$ will reach link l in time

$$\Delta_l^+ = \min\left\{ w_q^{min} : x_{p_i}(a+1) \neq x_{p_i}(a), p_i \underset{l}{\leftrightarrow} q, q \in Q_l \right\}$$

The waiting time required for each link is $\Delta_l = (\Delta^- - \Delta^+)^+$. For the whole network, we need to wait $\max_l \Delta_l$ before we can perform the next update.

3.9 Experiments

The proposed algorithm is implemented as an application through Python 2.7 on top of the POX controller [7]. Mininet 2.1.0 [4], which supports OpenFlow 1.0, is installed on Linux-based Netrunner 14 [5] as the testbed environment. The existing paths for each source-destination pair are selected from the K-shortest paths by Yen's algorithm [110] with the edge cost equals to one. The existing paths are established by adding VLAN tag matching action to the switches in the network. Since the group table actions and weighted bucket selection are only supported by OpenFlow 1.1 or higher version, unequal traffic splitting is realized by UDP sending port control. The UDP traffic for testing is generated by Iperf [3].

We verify the congestion behavior by measuring the queue length of the

virtual link interface at the port of the switch. Since the links in Mininet are implemented by Linux qdisc, they work as queues controlled by HTB (hierarchical token bucket). The queue of the virtual link interface contains packets that are in transit and packets that are waiting to be transmitted. If the queue length is less than the maximum number of packets that can be in transit, then all packets in the queue are in transit. Otherwise all packet in the queue above the maximum number are waiting to be transmitted.

The section consists of four parts. In the first part, Example 3.1 is realized to show the congestion resulting from different latencies of different path. A more complicated topology is constructed to simulate Example 3.2 in the second part. Two update sequences are compared to show that the least step update is not necessary the quickest. We verify the validity of the congestion-free constraints in the third part. The final part includes an example to illustrate the possibility of achieving quicker update with the help of timing information. *Note that since Iperf has a resolution of 0.5 seconds, the experiments are performed with inflated values for delays.*

3.9.1 Congestion Resulting from Propagation Delay

We simulate Example 3.1 and under the Mininet context, a and e are two hosts composing user 1, which has a traffic demand of 1 Mbps. All links have a propagation delay of 2 seconds and a capacity of 1 Mbps. The two available paths are $p_1 = (a, b, c, d, e)$ and $p_2 = (a, b, d, e)$. The initial configuration is $\{x_{p_1}(0) = 1, x_{p_2}(0) = 0\}$ (Mbps) and the target configuration is $\{y_{p_1} = 0, y_{p_2} = 1\}$ (Mbps). Consider the one-shot update, $\{x_{p_1}(1) = y_{p_1} = 0, x_{p_2}(1) = y_{p_2} = 1\}$. Since the

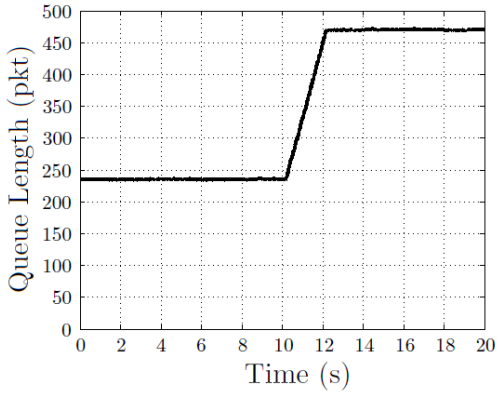
packets take a longer time to travel through p_1 than p_2, congestion will occur on the link (d, e) because of a mix of old traffic from p_1 and new traffic from p_2.

The simulation results are shown in Figure 3.3 and it supports the analysis above. One-shot update takes place at the 6$^{\text{th}}$ second and by the 10^{th} second, the queue length at the virtual link interface for link (d, e) builds up for 2 seconds (Figure 3.3a). For the next 4 seconds, the packets transmitted on link (d, e) will be an even mix from the old and new flows. At host e, the receiving rate of in-order packets (Iperf ignores out-of-order packets from calculation) starts to drop at the 12^{th} second (Figure 3.3b), which lasts for 4 seconds until all the old flows have cleared from the network.

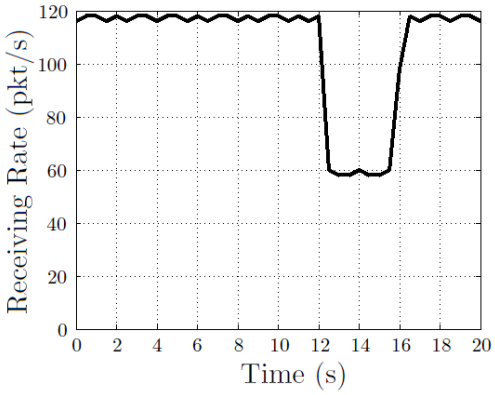
On the other hand, if user 1 reverse the process by shifting from path p_2 to p_1, congestion does not happen. One-shot update takes place at the 6$^{\text{th}}$ second, The simulation result in Figure 3.3c confirms the inference. When all the packets on path p_2 arrive, the new packets along path p_1 are still on the way. Thus the receiving rate measured on host h_2 drops to zero for 2 seconds, and the queue length decreases during the period when no packet comes in.

3.9.2 Update Steps and Update Time

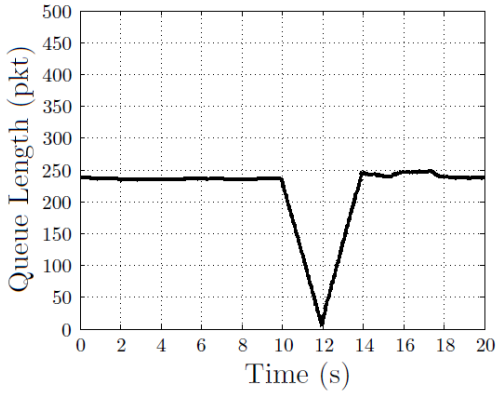
As discussed in Example 3.2, the shortest sequence of update steps does not imply the shortest update time. The link capacity are as specified in Example 3.2 while the propagation delay has been magnified by a hundred times. For user A, there are 4 paths available: $p_1 = (b, c, f, g)$,$p_2 = (b, f, g)$,$p_3 = (b, d, f, g)$ and $p_4 = (b, e, f, g)$. Similarly, 4 paths can be chosen for user B: $p_5 = (a, b, c, f, g)$,$p_6 = (a, b, f, g)$,$p_7 = (a, b, d, f, g)$ and $p_8 = (a, b, e, f, g)$.



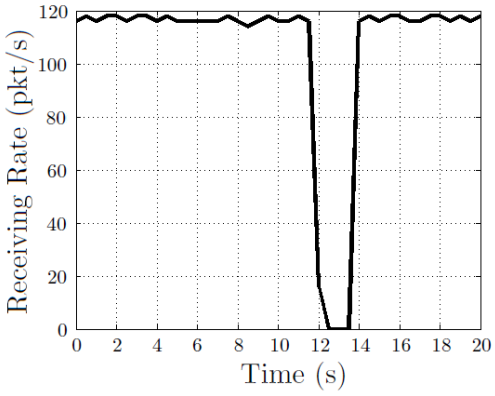
(a) Virtual queue length builds up for link (d, e)



(b) Receiving rate of in-order packets at host e

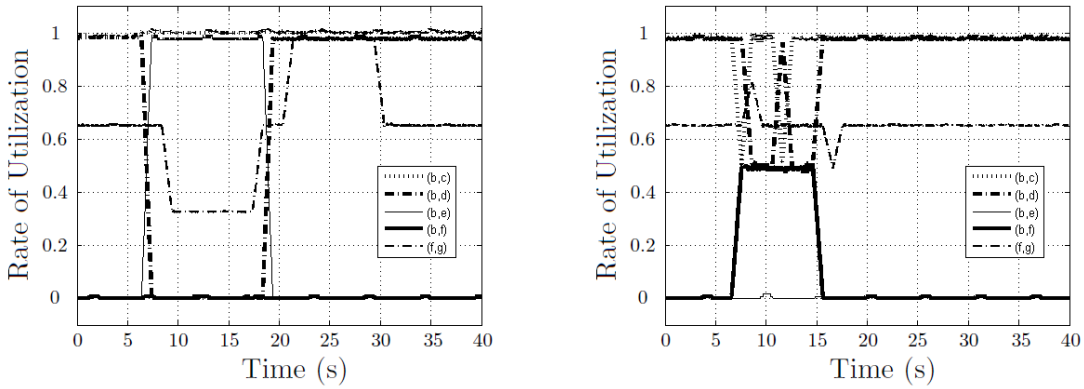


(c) Virtual queue length for link (d, e) clears



(d) No packets are received at e for 2 seconds

Figure 3.3: Simulation results for Example 3.1.



(a) Least step update

(b) Update with more steps but less time

Figure 3.4: The simulation results of Example 3.2

For notation simplicity, the configuration is expressed as a vector $v(a)$ where $v_i(a) = x_{p_i}(a)$ (Mbps) in the following context.

Now consider the initial configuration $(1,0,0,0,0,0,1,0)$ and the target configuration $(0,0,1,0,1,0,0,0)$. The congestion-free update sequence with least steps is given by $(1,0,0,0,0,0,1,0) \rightarrow (0,0,0,1,1,0,0,0) \rightarrow (0,0,1,0,1,0,0,0)$. The time-optimal solution is given by $(1,0,0,0,0,0,1,0) \rightarrow (.5,.5,0,0,.5,0,.5,0) \rightarrow (0,.5,.5,0,1,0,0,0) \rightarrow (0,0,1,0,1,0,0,0)$.

In Figure 3.4, the figure shows the rate of utilization for five key links. The rate of utilization of a link can be defined as the queue length of the virtual link interface divided by the maximum number of packets that can be transferred on the link. Congestion occurs if the rate of utilization exceed 1. Figure 3.4a shows least step update is congestion-free. Similarly, in Figure 3.4b, the time-optimal update sequence is also congestion-free, though it takes less time to achieve the target configuration.

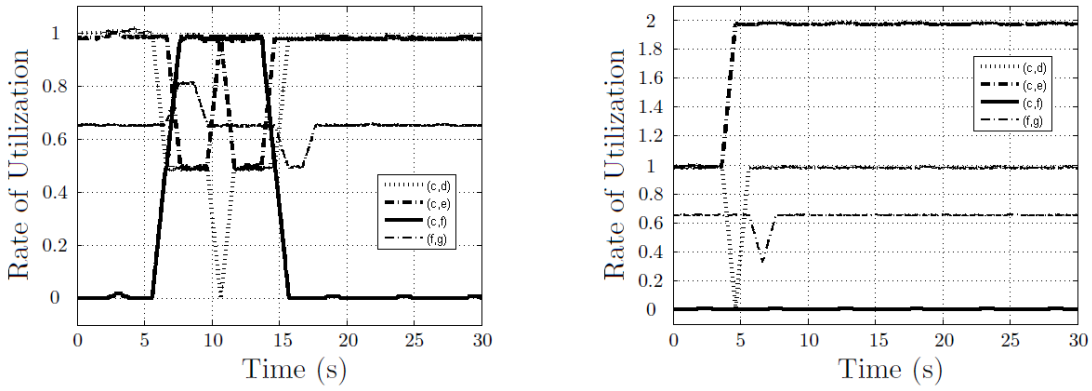
3.9.3 Timing Uncertainty and Congestion Constraints

In Example 3.3, the constraint set is formed by considering the time ranges. Here a simple example with the topology shown in Figure 3.5 is proposed to verify the validity of the constraint set. Suppose all the links have delay 1 (s). The capacity of the link (f, g) is 3 (Mbps). (a, c) and (b, c) have capacities large enough so that no congestion will occur on the links. All other inter-switch links have the capacity 1 (Mbps). Two users demand traffic 1 (Mbps) to g from a and b respectively.

The paths available for user 1 are $p_1 = (a, c, d, f, g)$,$p_2 = (a, c, f, g)$ and $p_3 = (a, c, e, f, g)$. p_4,p_5 and p_6 are defined similarly by replacing a with b on each path. The initial configuration and the target configuration are given by $(1, 0, 0, 0, 0, 1)$ and $(0, 0, 1, 1, 0, 0)$.

If there are timing uncertainties for both users, the congestion-free solution to OPT is $(1, 0, 0, 0, 0, 1) \rightarrow (.5, .5, 0, 0, .5, .5) \rightarrow (0, .5, .5, .5, .5, 0) \rightarrow (0, 0, 1, 1, 0, 0)$, which takes four steps. However, if both users are perfectly synchronized, one-shot update is feasible in this case.

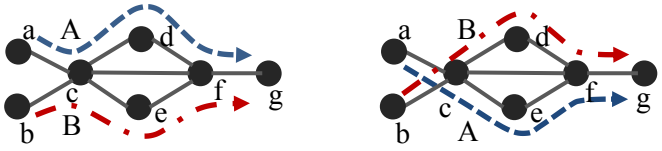
Figure 3.6 shows the results of the rate of utilization during the updates assuming a realized uncertainty of 1 second for user 2. We contrast what would happen if the users follow the 3-step congestion-free solution and if the users follow the one-shot update. The figures verify that the solutions to OPT are congestion-free, while a one-shot update would cause significant congestion.



(a) Optimal congestion-free update

(b) One-shot update

Figure 3.6: Realized uncertainty of 1 second delay for user 2



(a) Initial flow configuration

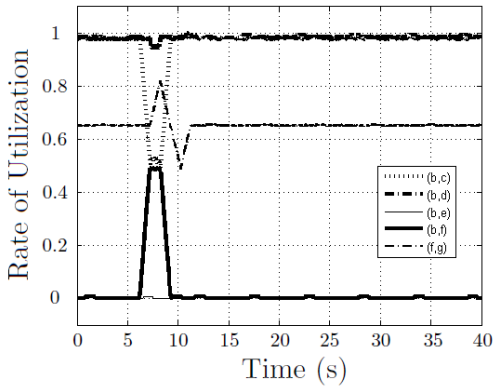
(b) Target flow configuration

Figure 3.5: Routing configuration for Section 3.9.3

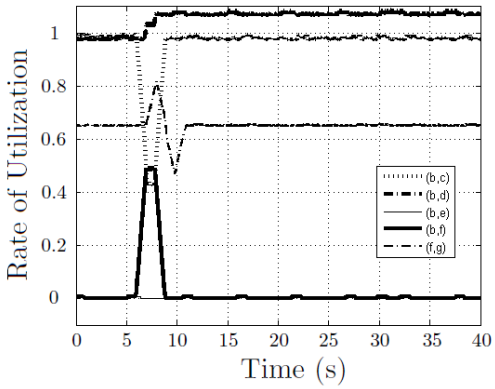
3.9.4 Achieving Shorter Update Time

As suggested in Example 3.4, it is possible to further shorten the update time by performing next update before the previous one has fully propagated through the network.

Example 3.4 is realized as we did in Section 3.9.2. The result in Figure 3.7a shows the possibility of further acceleration with the help of timing information. The key point here is that u_a is an upper bound to estimate the shortest time margin. By choosing a tighter bound $\max_l \Delta_l$ as the waiting time, it is possible to



(a) Updating without waiting the previous update fully cleared



(b) Congestion when the waiting time is less than $\max_l \Delta_l$

Figure 3.7: The simulation result of Example 3.4

achieve shorter update time without congestion. The bound $\max_l \Delta_l$ is a fairly tight bound to shorten the waiting time between steps for this case, as if we shorten the waiting time from $1 \to 1 \to 3$ to $.9 \to .9 \to 3$, congestion occurs as shown in Figure 3.7b.

CHAPTER 4

PACKET CLUSTERING INTRODUCED BY ROUTERS (NANOSECONDS - MICROSECONDS)

4.1 Introduction

The transition time or delay through a router for a packet can be divided into three components — transmission delay, queuing delay and processing delay. Transmission delay is the time to transmit all bits of a packet onto a data link, queuing delay is the time a packet spent in buffer while waiting for processing and transmission, and processing delay is the time to process the packet to determine the output port to transmit it. Transmission delay is deterministically determined by packet size and link capacity; queuing delay is variable as it depends on load, congestion and contention at the switching fabric; and processing delay is variable as tasks such as forwarding table lookup and quantization of packets into cells [24] could be variable. In general though, the variation in processing delay is assumed to be negligible and thus processing delay is usually modeled as deterministic.

To isolate the effect of processing delay from queuing delay, we consider an experimental setup where we send a stream of packets with fixed packet size and constant interpacket delay through an isolated and idle router with no cross traffic. Such an experiment was performed in [43] and the variation in processing time was reflected by the observation that the interpacket delay at the router's output exhibited variation on the order of 100 ns. More importantly, when the experiment was repeated for various interpacket delay constants, the variation was observed to be sufficient to induce packet clustering in some ex-

periments. Moreover, if the input stream went through multiple routers, the clustering effect were more prominent.

In this part of the thesis, we investigate the variation of processing delay and its effect on traffic behavior. Our focus here is twofold. First, we model and analyze the packet clustering induced by the variation in processing time. We propose a model for the transition time through a router and an accompanying metric to quantify packet clustering. Using the model, we provide analytical explanations for the various phenomena that was observed but not analyzed in [43]. Second, we investigate the possible causes for variation in processing time. Note that while white papers, for instance [1], exist to explain conceptually how a packet transitions through a commercial router, the actual implementation is proprietary. As such, our investigation focuses only on factors that could be experimentally controlled and validated. We examine the impact of packet sizes, clock drift, and forwarding table lookup on the variability of processing time. Such a fine-scale investigation was not feasible experimentally prior to [43], as network measurement devices have significant measurement error, see e.g. [72]. BIFOCALS as introduced in [43] achieved bit-level precision by directly capturing the physical layer symbol stream in real-time and time-stamping in off-line post-processing.

4.2 Related Work

For real-world network traffic, the observation that packets tend to cluster together or become bursty after passing through one or multiple routers is well-documented for several timescales [21, 17, 63, 52]. On longer timescales, pro-

posed explanations for burstiness are centered on input traffic characteristics such as the distribution of user's idle and active time [108], the distribution of file sizes [30], and TCP congestion control [52, 99]. At the packet level, the clustering could be attributed to contention and scheduling with cross-traffic at the switching fabric. In [43] though, the experiments are of a finer timescale and all the factors mentioned here are not applicable.

From a traffic engineering perspective, our work can affect all three requirements — delay, throughput and packet positioning. Clustering packets affects queuing delay [83], packet loss rate [72], and resource allocation [70], particularly for protocols involving pre-negotiated resources such as Diffserv [16]. Interpacket delay, on the other hand, has been used to detect congestion[20], estimate bandwidth with packet train [67], trace encrypted connections [107] and load balance traffic without packet reordering [55]. Lastly, interpacket delay variation, i.e. jitter is an important metric for multimedia and real time protocols such as RTP [94].

4.3 Motivation: Observed Phenomena

In this section, we motivate our study by presenting the key phenomena observed in the experiments of [43]. The data presented here are obtained by replicating the experiment using SoNIC [62]. Readers interested in further details should refer to either paper for more information. We first establish some terminologies and conventions before moving on to describe the various experiment setups and their associated observations.

We start with the basic measurement. The interpacket delay (IPD) is the

space, in bits, between the first bit of a packet and the first bit of the subsequent packet. The interpacket gap (IPG) is the space between the last bit of a packet and the first bit of the subsequent packet, i.e. IPG $=$ IPD $-$ packet size. Both IPD and IPG are measured on the physical layer of the 10 Gigabit Ethernet (10 GbE) and so to be precise, packet size from here on refers to the length of the Ethernet frame after preamble bits and control characters are inserted by 64/66b encoding (see Clause 49 of IEEE802.3 [2]). So while Ethernet frame sizes range from 72 to 1526 bytes, the actual packet size is longer after 64/66b encoding. Table 4.1 shows various Ethernet frame sizes and their corresponding packet sizes. Note that due to 64/66b encoding, the actual capacity of 10 GbE is $c = 10\,\text{Gbps} \times 66/64 = 10.3125\,\text{Gbps}$. For convenience, we often refer to time in units of bits instead, where 1 bit is equivalent to 97 ps (1/10.3125 Gbps). We say that a stream of packets is *homogeneous* if all the packets have the same size, the same interpacket delay, the same payload, and are heading for the same destination. We specify a homogeneous packet stream via two parameters: packet size, l and data rate, r. The associated interpacket delay could be figured out using the relation

$$\frac{l}{\text{IPD}} = \frac{r}{c} \tag{4.1}$$

We use shorthand such as 1526B 3G to refer to a homogenous packet stream with 1526-byte Ethernet frames and 3 Gbps data rate. As variation in processing time is inherent to a router, we refer to variation in processing time interchangeably with *inherent variation* of a router.

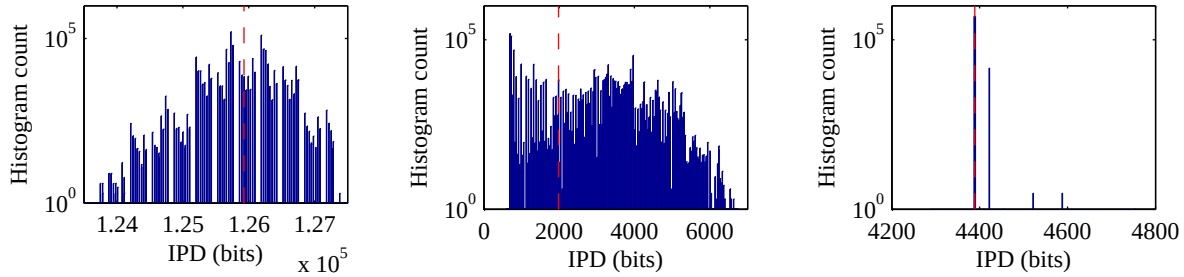
The first experiment of [43] sends a homogeneous packet stream through an isolated Cisco 6500 and the resulting IPD of the output stream is measured. We repeat the experiment and plot the IPD histogram in Figure 4.1a. The x-axis is the IPD measured in bits while the y-axis is the count of each IPD on a log-

Table 4.1: Interpacket delay and interpacket gap for various packet sizes and data rates. All values except for Ethernet frame size are measured from the physical layer of 10 Gigabit Ethernet.

Ethernet Frame Size [bytes]	After 64/66b Encoding [bits]	Nominal Data Rate [Gbps]	IPD [bits]	IPG [bits]
1526	12588	1	125928	113340
520	4288	6	7194	2906
72	594	3	1980	1386
72	594	6	990	396
72	594	9	660	66

scale. The router's inherent variation causes the actual histogram to be spread out. Further complications arise when the experiment is repeated with different data rates, packet size and number of routers. While the variation may seem small when the IPD is large, when the setup is varied by having a higher data rate and smaller packet size, the variation coupled with the small IPD is sufficient to induce packet clusters (see figure 4.1b, note that the highest count is at the leftmost value, where the IPG is minimal). The probability of observing such packet clusters further increases if the packets have to pass through multiple routers. In short, inherent variation is observed to induce packet clusters. However, when the input data rate is close to capacity, the effect of inherent variation seems to disappear and there is packet loss (Figure 4.1c).

Our experiments and analysis in the subsequent sections are motivated by Figure 4.1. We first establish the router model with inherent variation (Section 4.4.1) and quantify packet clustering (Section 4.4.2). Using the model, we show how IPD and degree of clustering changes as a packet stream passes through one router (Section 4.5), multiple routers (Section 4.6), and with cross traffic (Section 4.7). Finally, we physically validate the model with various routers (Section 4.9) and explore factors that could affect inherent variation (Section 4.9.3).



(a) 1526B 1G (b) 72B 3G (c) 520B 10G

Figure 4.1: Histogram of 1 million observed interpacket delay at the output of Cisco 6500 for various combination of data rates and packet size. The dotted red line marks the interpacket delay of the homogeneous packet stream. Note that for 520B 10G, the total does not tally up to 1 million as some packets are lost.

4.4 Modeling

4.4.1 Router Model

We assume the packet input stream is homogeneous with parameters (l, r); though all our results in this section hold as long as the IPD is independent and identically distributed (i.i.d.). Each packet of the input stream is indexed with i and transitions through a router by first experiencing processing delay followed by transmission delay. Each delay is modeled as a single server and thus the router is modeled as two servers in series (see figure 4.2). In general, the processing time may be correlated over time depending on the contents of a packet and the inner workings of a router. The first factor does not apply as the input stream is homogeneous. On the other hand, as we mentioned before, the actual workings of how a packet is processed is proprietary. As such, for tractability, we assume the processing time of each packet, X_i is i.i.d. As for the transmission time, it is simply the packet size divided by capacity rate, which we denote as $u \triangleq l/c$. This two-server model means that it is possible for paral-

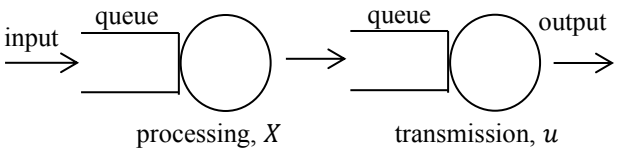


Figure 4.2: A two-server model of a router with a random processing time, X and a constant transmission time, u

lel processing and transmission of different packets, i.e. pipelining, to occur.

We denote the interpacket delay between packet i and $i+1$ with the random variable D_i. Since we are modeling actual packets going through the same input port, the interpacket delay in between packets must be larger than the time it takes a router to receive a packet, i.e. $D_i > u$. The router is capable of operating at capacity and as such packets must be processed faster than it can be transmitted, i.e. $u > E[X]$. We also assume there is sufficient buffer such that neither the processing server nor the transmission server ever overflows. In short, we have modeled the router with two serial servers with deterministic arrivals, independent service time and a first-in, first-out queue discipline. In the terminology of queuing theory, the processing server is a $D/G/1$ queue (**D**eterministic arrival/**G**eneral service time/**1** server) while the transmission server is a $G/D/1$ queue (Arrival with **G**eneral distribution/**D**eterministic service time/**1** server).

With this simple model, we are going to analyze how interpacket delay varies as the input stream passes through a router.. But first we clarify the notational convention of this paper. We use superscript on the variables to denote router number, not just on D, but also on S, X and other yet to be introduced variables, e.g. D_i^1 is the ith interpacket delay after going through the first router. The 0 superscript refers to the input packet stream. Since we will have expressions involving exponents, except for 0 or 1, all other integer superscripts should

be interpreted as exponents, e.g. D_i^2 is the square of the ith interpacket delay and not the ith interpacket delay after passing through 2 routers. The superscript is sometimes omitted when we are referring to interpacket delay in general while the subscript is sometimes omitted when the statement is applicable to all packets. We use the corresponding lowercase letter when a random variable takes on a particular value, e.g. $\Pr(D = d)$ is the probability of the random variable D taking on a value of d. We denote $E[\cdot]$ as the expectation function and any random variable with a tilde accent represents its zero-mean equivalent, e.g. $\tilde{X} = X - E[X]$.

4.4.2 Packet Clustering Metric

We propose a metric to represent the degree of packet clustering. To start, given that the packet arrival rate is smaller than the capacity rate, we know the average input data rate is the same as the average output data rate. This simple observation implies that the average IPD is the same before and after going through a router. Armed with this observation, we consider the simplest setup with a sequence of 3 packets as shown in Figure 4.3 to gain intuition. Since the average IPD is the same, the only variable here is the relative position of the second packet. Intuitively, the least clustering setup is when the packets are uniformly spaced, i.e. the IPG between packet 1 and 2 is the same as between packet 2 and 3. Packets are seen to be more clustered as packet 2 is closer to either packet 1 or 3 and the packets are most clustered when two packets have minimal IPG in between. As shown in Figure 4.3, a metric that fits the intuition is the sum of the squares of the IPG. For a homogeneous packet stream, the sum of squares of the IPD would fit the intuitive ordering as IPD $=$ IPG $+ l$. Simi-

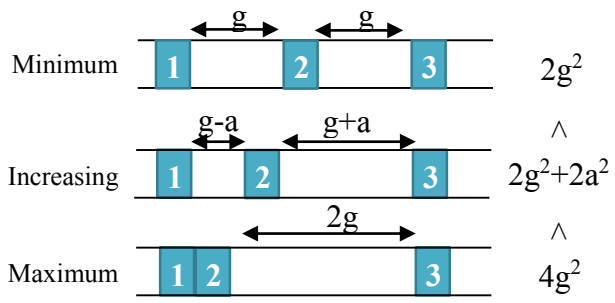


Figure 4.3: An intuitive figure depicting how packet clustering evolves as interpacket delay changes. The square of the interpacket gap fits the intuitive notion.

larly, the sample variance of the IPD would fit the intuitive ordering as it is the difference between the average sum of squares and average squared and this is the metric that we will be using, i.e. the packet clustering metric, C, given the IPD of all packets $\{d_i\}_{i=1}^{n}$ is given by

$$C(d) = \frac{1}{n-1}\sum_{i=1}^{n}\left(d_i - \bar{d}\right)^2 \tag{4.2}$$

where $\bar{d} = \sum_i d_i/n$ is the average IPD. Often times, we are dealing with random values D_i instead of a deterministically given d_i. In this case, the expected packet clustering is more relevant:

$$C(D) = \frac{1}{n}\sum_{i=1}^{n}\mathrm{var}(D_i) \tag{4.3}$$

When D_i is i.i.d., the expression simplifies to the variance of the interpacket delay. In short, the metric $C(D)$ states that a packet stream is more clustered if its IPD is more variable.

We note here that our proposed metric is not the only one that fits the intuitive notion of packet clustering in Figure 4.3. Other convex metrics are possible and we choose the current metric due to its relative ease of analysis and its apparent relation to inherent variation. Whenever possible, we will explain the results obtained in an intuitive manner without relying on our choice of metric.

4.5 The Single-Hop Case

We begin our analysis with the single hop case. There are two regions to consider here: when all packets have no waiting time (*large interpacket gap)* and when some of them possibly do (*small interpacket* gap). For a homogeneous packet stream, packets have no waiting time if it arrives after the prior packets have been transmitted from the router. A sufficient condition for this to occur is the interpacket delay being greater than the sum of processing and transmission time, i.e. $D^0 > X + u \iff G^0 > X$ holds with probability 1, where D^0 and G^0 are respectively, the IPD and the IPG of the input packet stream.

To illustrate, consider the parameter space (l, r) for a homogeneous packet stream. Denote g as the interpacket gap constant and x_{max} as the maximum processing time. Then from (4.1), we find

$$\frac{u}{g+u} = \frac{l}{c} \implies g > x_{max} \iff u\left(\frac{c-r}{r}\right) > x_{max} \tag{4.4}$$

The region surrounded by borders in Figure 4.4 represents all the feasible parameter combinations for a homogeneous packet stream. We divide it into the two regions according to (4.4) using $x_{max} = 2200$ bits and assuming x_{max} is the same for all packet sizes.

4.5.1 Large interpacket gap

With no waiting time, the interpacket delay after one router is relatively straightforward to figure out. Consider the sequence of discrete events that occur on packet 1 and 2 as shown in figure 4.5. Packet 1 arrives at the router, and takes

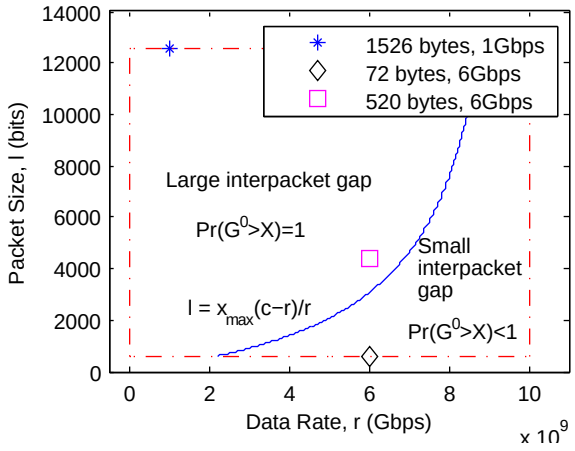


Figure 4.4: Large and small interpacket gap region for a router with $x_{max} = 2200$ bits for all packet sizes.

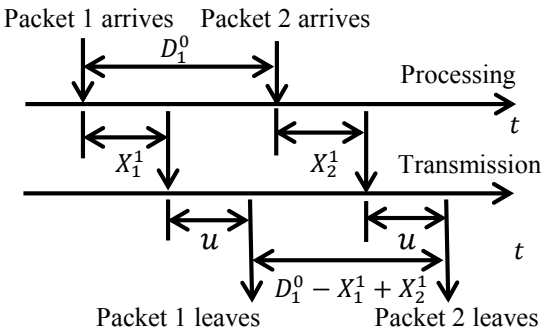


Figure 4.5: Large interpacket gap

a time period $X_1^1 + u$ to be processed and transmitted. The second packet then arrives and takes $X_2^1 + u$ to be processed and transmitted. The interpacket delay after 1 router for packet 1 and 2 is thus $D_1^1 = D_1^0 - X_1^1 + X_2^1$. One could continue by considering packets $3, 4, 5$ and so on to find that the same results hold:

$$D_i^1 = D_i^0 - X_i^1 + X_{i+1}^1,\ i = 1, 2, \ldots \tag{4.5}$$

While equation (4.5) looks simple, we can extract plenty of information: symmetry of interpacket histogram (Lemma 4.1 and Theorem 4.3), negative correlation of adjacent interpacket delay (Theorem 4.2) and an accurate unbiased estimator to represent a router's inherent variation (Lemma 4.4). Note that

while we have assumed a homogeneous input packet stream, all the results in this subsection hold as long as D^0 is i.i.d., symmetrical about $E\left[D^0\right]$ and satisfy $G^0 > X^1$.

To prove symmetry, we say that a random variable D has a distribution that is symmetrical about d if $D - d$ has the same distribution as $-(D - d)$, which we write as

$$D - d \sim -(D - d)$$

For such a random variable, its probability density function is symmetrical about d.

Lemma 4.1. *If the distribution of D^0 is symmetric about $E\left[D^0\right]$, then the distribution of $D_i^0 - X_i + X_{i+1}$ is symmetric about $E\left[D^0\right]$ for all i.*

Proof. D^0 is symmetric about $E\left[D^0\right]$ implies that

$$D^0 - E\left[D^0\right] \sim -\left(D^0 - E\left[D^0\right]\right) \tag{4.6}$$

Since the sequence $X_1^1, X_2^1, \ldots$ is i.i.d.,

$$-X_i^1 + X_{i+1}^1 \sim X_i^1 - X_{i+1}^1 \tag{4.7}$$

In general, if random variables K and L are independent, and similarly M and N are independent, and $K \sim M$ while $L \sim N$, then $K + L \sim M + N$. Thus, putting (4.6) and (4.7) together gives

$$D_i^0 - E\left[D^0\right] - X_i^1 + X_{i+1}^1 \sim -\left(D_i^0 - E\left[D^0\right] - X_i^1 + X_{i+1}^1\right)$$

which proves that the distribution of $D_i^0 - X_i + X_{i+1}$ is symmetric about $E\left[D^0\right]$.

□

However, Lemma 4.1 is not sufficient to prove symmetry, as the interpacket delay sequence $D_1^1, D_2^1, \ldots$ is not i.i.d. Neighboring terms of the sequence are correlated: $D_1^0 - X_1^1 + X_2^1$ is not independent of $D_2^0 - X_2^1 + X_3^1$ due to the X_2^1 term. Since the interpacket delays are correlated via the X_2^1 term, which has an opposite sign in each of the interpacket delay, we expect the correlation to be negative. In simple words, if the processing time of packet 2, X_2^1 is large, D_1^1 would be large while packet 3 will catch up to packet 2 making D_2^1 small and vice versa. We state the negative correlation formally.

Theorem 4.2. *After one hop, the correlation coefficient between two neighboring interpacket delay is -1/2, that is*

$$\rho_{i,j} \triangleq \frac{cov\left(D_i^1, D_j^1\right)}{\sqrt{var\left(D_i^1\right) var\left(D_j^1\right)}} = -\frac{1}{2}, \text{ for } |i-j| = 1 \tag{4.8}$$

Proof. The covariance of the interpacket delay is, by definition,

$$\begin{aligned} & \operatorname{cov}\left(D_i^1, D_j^1\right) \triangleq E\left[\tilde{D}_i^1 \tilde{D}_j^1\right] \\ = \; & E\left[\left(\tilde{D}_i^0 - \tilde{X}_i^1 + \tilde{X}_{i+1}^1\right)\left(\tilde{D}_j^0 - \tilde{X}_j^1 + \tilde{X}_{j+1}^1\right)\right] \end{aligned}$$

Expanding the equation gives us 9 terms. However, 5 terms involving either $\tilde{D}_i^0$ or $\tilde{D}_j^0$ is equal to zero since it is independent of all other terms. The independence implies, for instance, $E\left[\tilde{D}_i^0 \tilde{X}_{j+1}^1\right] = E\left[\tilde{D}_i^0\right] E\left[\tilde{X}_{j+1}^1\right] = 0$. For the other 4 terms, since the sequence $X_1^1, X_2^1, \ldots$ is i.i.d.,

$$E\left[\tilde{X}_i^1 \tilde{X}_j^1\right] = \begin{cases} E\left[\tilde{X}_i^1\right] E\left[\tilde{X}_j^1\right] = 0 & \text{, if } i \neq j \\ E\left[\left(\tilde{X}_i^1\right)^2\right] = \operatorname{var}\left(X^1\right) & \text{, if } i = j \end{cases}$$

Taking the sum of all 4 terms give

$$\text{cov}\left(D_i^1, D_j^1\right) = \begin{cases} 2\text{var}\left(X^1\right) & \text{, if } i = j \\ -\text{var}\left(X^1\right) & \text{, if } |i-j| = 1 \\ 0 & \text{, otherwise} \end{cases} \tag{4.9}$$

Applying equation (4.9) on the definition in equation (4.8) and using the fact that $\text{var}\left(D_i^1\right) = \text{cov}\left(D_i^1, D_i^1\right)$ give the desired answer. □

Even though the interpacket delay is negatively correlated, we can still prove symmetry by observing that every other term of the interpacket delay sequence is i.i.d and by using superposition.

Theorem 4.3. *The interpacket delay histogram of a homogeneous packet stream at a router's output is symmetrical about* $E\left[D^0\right]$.

Proof. Consider the first and third IPD: $D_1^1 = D_1^0 - X_1^1 + X_2^1$ and $D_3^1 = D_3^0 - X_3^1 + X_4^1$. Since $X_1^1, X_2^1, ...$ are i.i.d. while D^0 is just a constant for a homogeneous packet stream, D_1^1 and D_3^1 are i.i.d. of each other. We can easily generalize this and find that the odd-indexed sequence $D_1^1, D_3^1, \ldots$ is i.i.d., and so is the even-indexed sequence $D_2^1, D_4^1, \ldots$. From Lemma 4.1, we know that the IPD histogram of either the odd or the even-indexed sequence is symmetrical about $E\left[D^0\right]$. The IPD histogram of the whole sequence is just the superposition of the two, and is hence symmetrical about $E\left[D^0\right]$. □

We now switch attention to packet clustering. Packet clustering could be determined using equation (4.5). Since D_i^0, X_i^1 and X_{i+1}^1 are all independent of each other

$$C\left(D^1\right) = \text{var}\left(D^0\right) + 2\text{var}\left(X^1\right) \tag{4.10}$$

For a homogeneous input packet stream, D^0 is a constant and thus $C\left(D^1\right) = 2\text{var}\left(X^1\right)$. The variance of the processing time $\text{var}\left(X^1\right)$ is an important expression that will continue to reappear for the rest of the paper. From a router performance evaluation standpoint, we could think of $\text{var}\left(X^1\right)$ as either the metric or the benchmark characterizing the inherent variation of a particular router model. Thus, for practical purposes, it is important to be able to estimate it from available data as accurately as possible. Keeping in mind the negative correlation of the IPD, we could estimate the variance by applying the unbiased sample variance formula on the i.i.d. odd and evenly indexed IPD and then average the two estimates. However, it turns out that despite the negative correlation, we could make an unbiased estimation from the full IPD sequence.

Lemma 4.4. *Given $D_1^1, D_2^1, \ldots, D_n^1$, the sequence of interpacket delays from the output of a router fed with a homogeneous input packet stream, the estimator*

$$\beta = \frac{n}{(n+1)(n-1)} \sum_{i=1}^{n} \left(D_i^1 - \bar{D}\right)^2$$

where $\bar{D} = \frac{1}{n}\sum_{i=1}^{n} D_i^1$ is the sample mean, is an unbiased estimator of $2var\left(X^1\right)$, i.e. $E[\beta] = 2var\left(X^1\right)$.

Proof.

$$\begin{aligned} n(n+1)(n-1)E[\beta] &= E\left[\left(n\tilde{D}_i - \sum_{j=1}^{n}\tilde{D}_j^1\right)^2\right] \\ &= \text{var}(X)\cdot\sum_{i=1}^{n}\left[2n^2 - 4n + 2n\sum_{|i-j|=1} 1 + 2n - 2(n-1)\right] \\ &= 2n\text{var}(X)\left[n^2 - 2n + 2(n-1) + n - (n-1)\right] \\ &= 2n(n+1)(n-1)\,\text{var}(X) \end{aligned}$$

where the third equality follows from applying correlation values from Theorem 4.2 and noting that if we expand out $\left(\sum_{j=1}^{n} \tilde{D}_j^1\right)\left(\sum_{k=1}^{n} \tilde{D}_k^1\right)$, there are $2\,(n-1)$ terms where $|j-k|=1$. □

4.5.2 Small interpacket gap

The interpacket gap in this region is small while the service time is sometimes long enough to induce a waiting time in the next packet. The IPD histogram is no longer symmetrical as its left portion is distorted by the physical requirement of a non-negative interpacket gap (Figure 4.1b). To get a clearer picture and to figure out how packet clustering happens, we follow the development of the previous subsection by considering the sequence of discrete events that occur on packet i and $i+1$. It turns out that we need to consider 3 general cases: positive idle time (figure 4.6a), zero idle time (figure 4.6b), and positive waiting time at the transmission server (figure 4.7). We define I_i as the processing server idle time in between packet i and $i+1$ and W_i as the waiting time of packet i at the processing server.

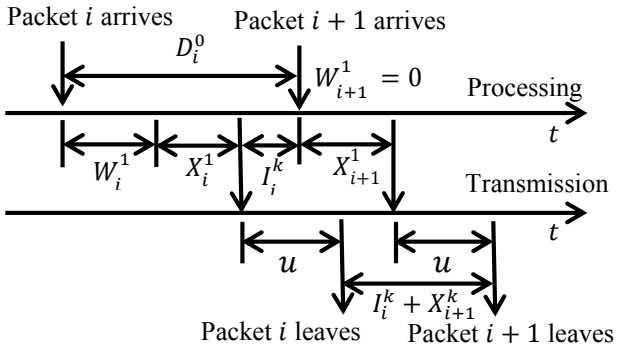
From figure 4.6a and 4.6b, we see that the idle time is given by

$$I_i^1 = \max\left(0, D_i^0 - W_i^1 - X_i^1\right) = \left(D_i^0 - W_i^1 - X_i^1\right)^+ \tag{4.11}$$

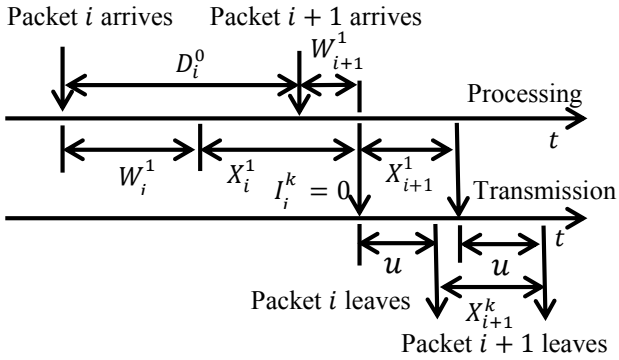
where $a^+ = \max(0, a)$. Similarly, the waiting time of packet $i+1$ is given by

$$W_{i+1}^1 = \max\left(0, W_i^1 + X_i^1 - D_i^0\right) = \left(D_i^0 - W_i^1 - X_i^1\right)^- \tag{4.12}$$

where $a^- = \max(0, -a)$. In the queuing theory literature, equation (4.12) is also known as the Lindley equation [66]. Combining all three cases, the IPD



(a) IPD when there is positive idle time



(b) IPD when there is zero idle time

Figure 4.6: Two possible sequence of events in between the arrivals and departures of packets i and $i+1$

after one-hop is given by

$$D_i^1 = \max\left\{u, I_i^1 + X_{i+1}^1\right\} \tag{4.13}$$

The identity $a = a^+ - a^-$ gives us

$$D_i^0 - W_i^1 - X_i^1 = I_i^1 - W_{i+1}^1 \tag{4.14}$$

Combining equation (4.13) and (4.14) gives

$$D_i^1 \;\geq\; D_i^0 - \left(W_i^1 + X_i^1\right) + \left(W_{i+1}^1 + X_{i+1}^1\right) \tag{4.15}$$

Recall that for the large interpacket gap region, equation (4.10) tells us that packet clustering increases by 2var $\left(X^1\right)$. For the small interpacket gap region,

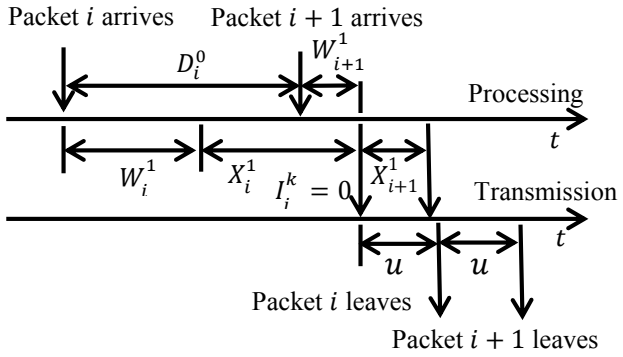


Figure 4.7: IPD when there is waiting time at the transmission server. The idle time could be positive as in Figure 4.6a.

some packets are prevented from getting closer together due to the physical requirement of a minimum service time. This implies that intuitively, packet clustering after one router should increase to a value that is less than 2var $\left(X^1\right)$. Before determining the exact value, we need a supporting lemma.

Lemma 4.5. *For any random variable* $Y = Y^+ - Y^-$, $var\left(Y\right) = var\left(Y^+\right) + var\left(Y^-\right) + 2E\left[Y^+\right]E\left[Y^-\right]$.

Theorem 4.6. *For a packet stream with i.i.d. interpacket delay and* $n \to \infty$ *number of packets, after one hop,* $C_v\left(D^1\right)$ *is given by*

$$var\left(D^1\right) \leq var\left(D^0\right) + 2\left(var\left(X^1\right) - E\left[W^1\right]E\left[I^1\right]\right) \tag{4.16}$$

Proof. We start with equation (4.13) and noting that forcing a minimum value reduces variance

$$\text{var}\left(D_i^1\right) \leq \text{var}\left(I_i^1 + X_{i+1}^1\right) = \text{var}\left(I_i^1\right) + \text{var}\left(X_{i+1}^1\right) \tag{4.17}$$

where the second equality follows from independence of I_i^1 and X_{i+1}^1. We look for an expression to substitute for var $\left(I_i^1\right)$ by applying lemma 4.5 to $Y = D_i^0 - W_i^1 - S_i^1$.

$$\text{var}\left(D_i^0 - W_i^1 - X_i^1\right) \;=\; \text{var}\left(I_i^1\right) + \text{var}\left(W_{i+1}^1\right) + 2E\left[I_i^1\right]E\left[W_{i+1}^1\right] \tag{4.18}$$

Since X_i^1 is independent of D_i^0 and W_i^1, and D_i^0 is independent of W_i^1, LHS of (4.18) is equal to $\text{var}\left(D_i^0\right) + \text{var}\left(W_i^1\right) + \text{var}\left(X_i^1\right)$. Rearranging the terms, we obtain an expression for $\text{var}\left(I_i^1\right)$ and substitute it into (4.17) to find

$$\text{var}\left(D_i^1\right) \;\leq\; \text{var}\left(D_i^0\right) + \text{var}\left(W_i^1\right) + 2\text{var}\left(X^1\right) - \text{var}\left(W_{i+1}^1\right) - 2E\left[I_i^1\right] E\left[W_{i+1}^1\right]$$

As $n \to \infty$, each term converges to equilibrium values, e.g. $D_i^1 \to D^1$, $I_i^1 \to I^1$ and $W_{i+1}^1 \to W^1$. We obtain

$$\text{var}\left(D^1\right) \leq \text{var}\left(D^0\right) + 2\left(\text{var}\left(X^1\right) - E\left[W^1\right] E\left[I^1\right]\right) \tag{4.19}$$

as desired. □

Since $E\left[W^1\right] E\left[I^1\right] > 0$, Theorem 4.6 confirms our intuition that packet clustering increases by a value less than $2\text{var}\left(X\right)$. The theorem also tells us that it is possible for the packet stream to be less clustered if the input packet stream is not homogeneous and $E\left[W^1\right] E\left[I^1\right] > \text{var}\left(X\right)$. However, we may not be able to check for it in practice, since we may not have sufficient knowledge or access to determine $E\left[W^1\right]$ and $E\left[I^1\right]$. We thus have the question: for an input packet stream with i.i.d. IPD, is there an easily verifiable condition to know when the packet stream will be less clustered?

Corollary 4.7. *An input packet stream with i.i.d. interpacket delay will be less clustered after going through a router if* $2var\left(X^1\right) \leq E\left[\left\{\left(D^0 - X^1\right)^-\right\}^2\right]$.

Proof. We apply a lower bound by Kingman [60]

$$2E\left[I^1\right] E\left[W^1\right] \geq E\left[\left\{\left(D^0 - X^1\right)^-\right\}^2\right]$$

If $2\text{var}\left(X^1\right) \leq E\left[\left\{\left(D^0 - X^1\right)^-\right\}^2\right]$, then the lower bound and (4.16) gives

$$\begin{aligned}\text{var}\left(D^1\right) &\leq \text{var}\left(D^0\right) + 2\text{var}\left(X^1\right) - E\left[\left\{\left(D^0 - X^1\right)^-\right\}^2\right] \\ &\leq \text{var}\left(D^0\right)\end{aligned}$$

i.e. packet clustering has decreased after passing through a router □

Note that $E\left[\left\{\left(D^0 - X^1\right)^-\right\}^2\right]$ is large if D^0 has a large probability of taking small values, i.e. a large portion of packets are close together. There are two scenarios for the packet to be closer together: one, the input packet stream is getting more clustered and two, the input data rate is getting higher, as higher data rate implies smaller interpacket delay. Since $\text{var}\left(D^0\right) + 2\text{var}\left(X^1\right) - E\left[\left\{\left(D^0 - X^1\right)^-\right\}^2\right]$ is an upper bound for $\text{var}\left(D^1\right)$, the upper bound is getting smaller as either scenario occurs. As such, it is reasonable to expect that *the change in packet clustering, var* $\left(D^1\right) - var\left(D^0\right)$ *to decrease as the input packet stream is getting more clustered or as the input data rate increases* (see Figure 4.11b). In cases where the input packet stream is very clustered, we have shown in Corollary 4.7 that the change in packet clustering could in fact be negative (see Figure 4.10b)

4.6 The Multi-Hop Case

In this section, we extend our results to the multiple-router scenario. We denote m as the number of routers and the routers are not assumed to be identical unless otherwise stated. There are now 3 regions to consider. Similar to before, the regions depend on the interpacket gap and maximum processing time. If all packets have zero waiting time while passing through all routers, i.e.

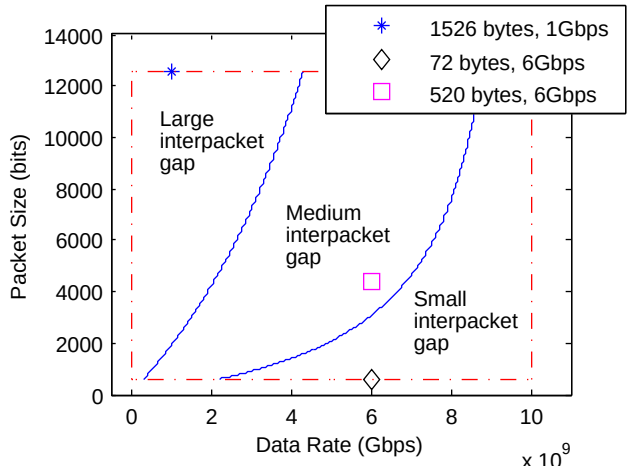


Figure 4.8: Large, medium and small interpacket gap region for $m = 8$ identical routers and $x_{max} = 2200$ bits

$G^0 > \sum_{k=1}^{m} x_{max}^k$, then we are in the large interpacket gap region. If all packets have zero waiting time for the first j routers and some packets have positive waiting time for the remaining routers, i.e. $G^0 > \sum_{k=1}^{l} x_{max}^k$ for all $l \leq j < m$ while for $m \geq l > j$, $G^0 < \sum_{k=1}^{l} x_{max}^k$ with positive probability, then we are in the medium interpacket gap region. In the small interpacket gap region, some packet have positive waiting time while passing through each router, i.e. $G^0 < \sum_{k=1}^{l} x_{max}^k$ with positive probability for $l = 1, \ldots, m$.

For $m = 8$ identical routers, the large, medium and small interpacket gap region are shown in Figure 4.8. With reference to the single-hop Figure 4.4, the small interpacket gap region remains the same while the large interpacket gap region is now divided into two. As the number of router grows, the medium interpacket gap region grows while the large interpacket gap region shrinks.

4.6.1 Large interpacket gap

The analysis is this subsection is mostly a straightforward generalization of the results of the single-hop case. Note that for this subsection only, the integer exponent refers to router number. First consider $m = 2$, since we are in the large interpacket gap region, equation (4.5) holds and we have

$$D_i^2 = D_i^1 - X_i^2 + X_{i+1}^2$$

Applying equation (4.5) again to D_i^1 gives

$$D_i^2 = D_i^0 - \left(X_i^1 + X_i^2\right) + \left(X_{i+1}^1 + X_{i+1}^2\right)$$

Generalizing, we obtain for all $l \leq m$

$$D_i^l = D_i^0 - \sum_{k=1}^{l} X_i^k + \sum_{k=1}^{l} X_{i+1}^k \tag{4.20}$$

The other results concerning negative correlation and symmetry of the IPD histogram generalize in the same manner. In addition, we have a new result concerning how packet clustering changes as it passes through the routers. We state them all formally in the following theorem.

Theorem 4.8. *In the large interpacket gap region, for any $1 \leq l \leq m$, the interpacket delay sequence $D_1^l, D_2^l, \ldots$ has the following 3 properties: adjacent interpacket delay has a correlation coefficient of $-1/2$,* the IPD histogram at the final router's output is *symmetrical and if the m routers are identical, then packet clustering increases linearly with the number of routers.*

Proof. To prove the first two properties, denote $Y_i = \sum_{k=1}^{l} X_i^k$ to get

$$D_i^l = D_i^0 - Y_i + Y_{i+1}$$

This is exactly equation (4.5) and since the sequence $Y_1, Y_2, \ldots$ is i.i.d., negative correlation and symmetry follows from Theorem 4.2 and 4.3. Finally,

$$\text{var}\left(D_i^l\right) \;=\; \text{var}\left(D^0\right) + 2\sum_{k=1}^{l} \text{var}\left(X^k\right)$$

For a homogeneous packet stream and identical routers, $\text{var}\left(D_i^l\right) = 2l\text{var}\left(X\right)$ and thus $C_v\left(D^l\right) = 2l\text{var}\left(X\right)$ □

4.6.2 Medium and small interpacket gap

For medium interpacket gap, for the first j routers where there is no waiting time, packet clustering increases linearly and for the remaining routers, the change in packet clustering would be similar to the small interpacket gap region, which we will analyze now.

For small interpacket gap, the results are not as easy to generalize from the single hop case as large interpacket gap. The main difficulty arises from the correlated interpacket delay after the first hop. Such correlation implies that the setup may not converge to a steady state distribution even if the input packet stream is sufficiently long. Additionally, even if the setup does converge, we could go through the same steps as the proof of Theorem 4.6 to find that due to correlation, we now have an additional covariance term, $\text{cov}\left(D^0, W\right)$ which does not have a clear interpretation. As such, we approximate by ignoring the correlation and assuming that the input to all the routers have i.i.d. interpacket delay and our aim in this subsection is not to derive analytical expressions but to use the results from Section 4.5.2 to argue qualitatively how packet clustering would evolve with increasing number of identical hops.

Recall that after Corollary 4.7 we argue that the change in packet clustering, var $\left(D^1\right) -$ var $\left(D^0\right)$ would decrease as the input packet stream is getting more clustered or as the input data rate increases. Given the i.i.d. input assumption to all routers, we could now apply this argument at each hop. This means that if we start with a homogeneous input packet stream, then packet clustering would be an increasing and concave function in the number of identical hops. Since var $\left(D^1\right) -$ var $\left(D^0\right)$ could be negative when packet clustering exceeds the threshold specified in Corollary 4.7, we expect packet clustering to converge to a value for sufficiently large number of identical hops. Conversely, if we start out with a very clustered input packet stream, then packet clustering would be a decreasing and convex function in the number of identical hops, and would also converge to a fix value for sufficiently large number of identical hops (see Figure 4.10b).

The second implication is that if we have two input packet streams with equal packet clustering but different data rate, then for the input stream with a higher data rate, its rate of change for packet clustering in the number of identical hops would be lower. For instance, suppose we have two homogeneous packet stream with 4G and 6G data rate. Then while packet clustering packet clustering would evolve in an increasing and concave manner in the number of identical hops for both, the function for 4G would be strictly above that of 6G's (see Figure 4.11b).

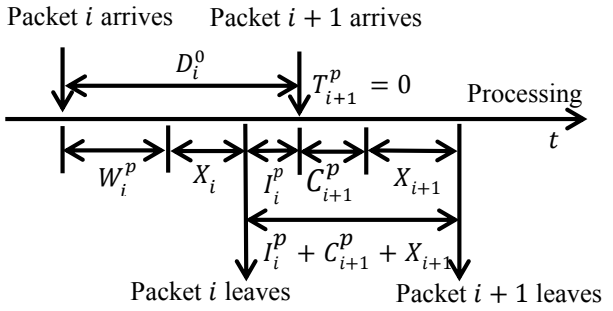
4.7 Adding Cross Traffic

In this section, we add in cross traffic and show that the results and discussions from Section 4.5 carry over. We need new terms and assumptions with the addition of cross traffic. We call the packet stream that we are tracking as it goes through the router the *target traffic* while any other traffic that interferes with it, the *cross traffic*. It is well-known that Internet traffic is usually not Poisson in nature and is positively correlated over time [63]. For such cross traffic, the setup that we are currently analyzing does not necessarily converge to the steady state distribution. To make the analysis tractable, we make the assumption that the cross traffic is not time varying, i.e. stationary, and the current setup does converge to the steady state distribution. We assume packets are served in a first-in, first-out manner. This is again a simplifying assumption as packets that arrive at different router input ports typically have to undergo contention and scheduling and the final packet service order is not necessarily first-in, first-out.

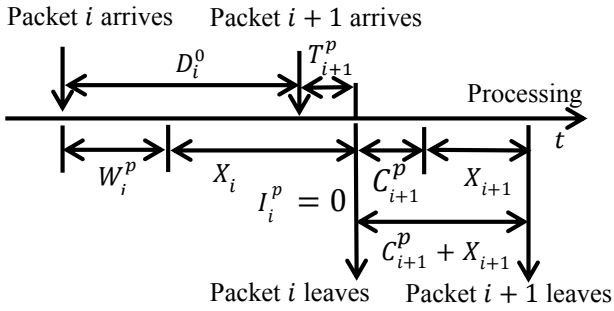
There are two key steps to analyzing the interaction of the two types of traffic. The first step is to divide our analysis into two stages by first finding the interpacket delay after passing through the processing server, Δ^1, before moving on to deal with the transmission server. The second step is to further subdivide the waiting time and idle time by source. The waiting time of the ith target packet at the processing server is now

$$W_i^p = T_i^p + C_i^p \tag{4.21}$$

where T_i^p is the waiting time incurred on the ith target packet till target packet $i-1$ is processed and C_i^p is the waiting time incurred on the ith target packet due to cross traffic that is processed after target packet $i-1$. We define I_i^p as the idle time the processing server spends not processing any target traffic after



(a) Positive idle time $I_i^p > 0$



(b) Zero idle time $I_i^p = 0$

Figure 4.9: Two possible sequence of events at the processing server in between the arrivals and departures of target packets i and $i+1$

the departure of the ith packet and before the arrival of the $(i+1)$th packet. Note that it is possible for the processing server to be processing cross traffic during such idle time. The terms W_i^r, T_i^r, C_i^r and I_i^r are defined analogously for the transmission server. To prevent notational clustering, we will drop the superscript 1 from all non interpacket delay terms in this section.

The analysis to derive the interpacket delay and packet clustering follow the same path as the model with no cross traffic. There are two general cases as

shown in Figure 4.9. Going through the same steps as before, we arrive at

$$\begin{aligned} I_i^p &= \left(D_i^0 - X_i - W_i^p\right)^+ && (4.22)\\ T_{i+1}^p &= \left(D_i^0 - X_i - W_i^p\right)^- && (4.23)\\ \Delta_i^1 &= I_i^p + C_{i+1}^p + X_{i+1} && (4.24)\\ D_i^0 - X_i - W_i^p &= I_i^p - T_{i+1}^p && (4.25)\\ \Delta_i^1 &= D_i^0 - W_i^p - X_i + W_{i+1}^p + X_{i+1} && (4.26) \end{aligned}$$

We next find out the change in packet clustering.

Lemma 4.9. *For a target packet stream with i.i.d. interpacket delay and $n \to \infty$ number of packets, if the current setup with cross traffic converges, then*

$$var\left(\Delta^1\right) = var\left(D^0\right) + 2\left[var\left(X\right) + cov\left(I^p + C^p, W^p\right)\right] \tag{4.27}$$

Proof. The proof is almost identical to the proof for Theorem 4.6 and thus some details some will be skipped. We start with equation (4.24)

$$\text{var}\left(\Delta_i^1\right) = \text{var}\left(I_i^p\right) + \text{var}\left(C_{i+1}^p\right) + \text{var}\left(X\right) + 2\text{cov}\left(I_i^p, C_{i+1}^p\right) \tag{4.28}$$

We will find an expression for var $\left(I_i^p\right)$ by applying lemma 4.5 together with equation (4.22) and (4.23)

$$\begin{aligned} &\text{var}\left(D_i^0\right) + \text{var}\left(X\right) + \text{var}\left(W_i^p\right)\\ =\ &\text{var}\left(I_i^p\right) + \text{var}\left(T_{i+1}^p\right) + 2E\left[I_i^p\right]E\left[T_{i+1}^p\right] && (4.29) \end{aligned}$$

Rearranging, substituting the expression for var $\left(I_i^p\right)$ into equation (4.28) and taking the limit as $i \to \infty$ where each term converges to equilibrium distribution, we find

$$\begin{aligned} \text{var}\left(\Delta^1\right) = \text{var}\left(D^0\right) &+ 2\text{var}\left(X\right) - 2E\left[I^p\right]E\left[T^p\right] + \text{var}\left(W^p\right)\\ &- \text{var}\left(T^p\right) + \text{var}\left(C^p\right) + 2\text{cov}\left(I^p, C^p\right) && (4.30) \end{aligned}$$

Since $\text{var}\,(W^p) - \text{var}\,(T^p) = \text{var}\,(C^p) + 2\text{cov}\,(T^p, C^p)$,$-E\,[I^p]\,E\,[T^p] = \text{cov}\,(I^p, T^p)$ and $\text{var}\,(C^p) = \text{cov}\,(C^p, C^p)$, the last 5 terms on the RHS of equation (4.30) simplifies to $2\text{cov}\,(I^p + C^p, W^p)$ and thus we are done. □

We then move on to the transmission server. Now note that the analysis is identical to the two cases shown in Figure 4.9, with Δ^1 replacing D^0, u replacing X_i and X_{i+1}, and waiting and idle time terms for the transmission server replacing analogous terms for the processing server.

Theorem 4.10. *Assuming the setup converges to equilibrium distributions, then*

$$var\left(D^1\right) \;\leq\; var\left(D^0\right) + 2var\,(X) + 2cov\,(I^p + C^p, W^p) + 2cov\,(I^r + C^r, W^r)$$

Proof. All the steps are the same as the proof of Lemma 4.9, except for equation (4.29) where we have an additional covariance term, $2\text{cov}\left(\Delta_i^1, W_i^r\right)$ due to the correlation of Δ_i^1 with adjacent interpacket delay values. Substituting Δ_i^1 using equation (4.26) and noting that all the terms are independent of W_i^r except W_i^p, we arrive at

$$2\text{cov}\left(\Delta_i^1, W_i^r\right) = -2\text{cov}\left(W_i^p, W_i^r\right) < 0$$

As we assume packets are processed at a faster rate than they could be transmitted, a larger waiting time at the processing server implies that we should expect a larger waiting time at the transmission server and thus $\text{cov}\left(W_i^p, W_i^r\right) > 0$. For the proof here, the equality of equation (4.29) is thus replaced with $\geq$, which carries through to give

$$\text{var}\left(D^1\right) \leq \text{var}\left(\Delta^1\right) + 2\text{cov}\,(I^r + C^r, W^r) \tag{4.31}$$

Now apply equation (4.9) and we are done. As a simple check, when there is no cross traffic, $C^p = 0$, $C^r = 0$, $I^p = I$, $T^p = W$, $\text{cov}\,(I, W) = -E\,[I]\,E\,[W]$,

$\text{cov}\left(I^r, T^r\right) = -E\left[I^r\right] E\left[T^r\right] < 0$ and the expression degenerates to equation (4.16). □

Before showing that packet clustering can decrease, we backtrack to the large interpacket gap scenario. From equation (4.26) and its analogous version for the transmission server, we have

$$D^1 = D_i^0 - W_i^p - W_i^r - X_i + W_{i+1}^p + W_{i+1}^r + X_{i+1} \tag{4.32}$$

Consider what happens when the target traffic is homogeneous with a sufficiently large interpacket gap. The waiting time is mostly due to cross traffic, i.e. $W_i^p \approx C_i^p$ and $W_i^r \approx C_i^r$. Since the cross traffic is stationary, C_i^p is approximately independent of and has similar distribution as C_{i+1}^p and similarly for C_i^r and C_{i+1}^r. In such a scenario we could apply the same reasoning as Section 4.5.1 to find that at the output, the target traffic has interpacket delays with adjacent values having a correlation of $-1/2$ and with a histogram that is symmetrical in shape (see Section 4.9.2).

We now show that even with cross traffic, it is still possible for packet clustering to decrease. Theorem 4.7 tells us that packet clustering could decrease when the input stream is already clustered. With the addition of cross traffic, we should expect the decrease could only happen when the input stream is more clustered than is required for the no cross traffic scenario.

Theorem 4.11. *With cross traffic, the change in packet clustering is upper bounded by*

$$
\begin{aligned}
& var\left(D^1\right) - var\left(D^0\right) \\
\leq\ & 2\left[var\left(X\right) + var\left(C^p\right) + cov\left(I^p, C^p\right) + var\left(C^r\right) + cov\left(I^r, C^r\right)\right] \\
& - E\left[\left\{\left(D^0 - X - C^p\right)^-\right\}^2\right] - E\left[\left\{\left(\Delta^1 - u - C^r\right)^-\right\}^2\right] \qquad (4.33)
\end{aligned}
$$

We need the help of the following lemma before proving.

Lemma 4.12. *At steady state*

(i) $E\left[I^p\right] = E\left[D^0 - X - W^p\right]$

(ii) $E\left[\left\{I^p\right\}^2\right] \leq E\left[\left\{\left(D^0 - X - C^p\right)^+\right\}^2\right]$

(iii) $\text{var}\left(I^p\right)$ is upper bounded by

$$\text{var}\left(D^0\right) + \text{var}\left(X\right) + \text{var}\left(C^p\right) - E\left[\left\{\left(D^0 - X - C^p\right)^-\right\}^2\right]$$

(iv) $\text{var}\left(I^r\right)$ is upper bounded by

$$\text{var}\left(\Delta^1\right) + \text{var}\left(C^r\right) - E\left[\left\{\left(\Delta^1 - u - C^r\right)^-\right\}^2\right]$$

Proof. The first expression could be obtained by taking the expectation on equation (4.24), apply $E\left[\Delta^1\right] = E\left[D^0\right]$ and rearrange the terms. As for the second equation, start from the inequality $\left(D^0 - X - W^p\right)^+ \leq \left(D^0 - X - C^p\right)^+$, square both sides, take expectation and apply equation (4.22). The upper bound for $\text{var}\left(I^p\right)$ is derived by applying the 2 expressions we just derived.

$$
\begin{aligned}
\text{var}\left(I^p\right) &= E\left[\left\{I^p\right\}^2\right] - E\left[I^p\right]^2 \\
&\leq E\left[\left\{\left(D^0 - X - C^p\right)^+\right\}^2\right] - E\left[D^0 - X - C^p\right]^2 \\
&= E\left[\left(D^0 - X - C^p\right)^2 - \left\{\left(D^0 - X - C^p\right)^-\right\}^2\right] - E\left[D^0 - X - C^p\right]^2 \\
&= \text{var}\left(D^0 - X - C^p\right) - E\left[\left\{\left(D^0 - X - C^p\right)^-\right\}^2\right]
\end{aligned}
$$

Expand out the first term and we are done. The upper bound for var (I^r) is derived in a similar manner. □

We now prove Theorem 4.11.

Proof. Start from $\Delta^1 = I^p + X + C^p$, take the variance, and apply the upper bound for var (I^p) from Lemma 4.12 to obtain

$$\begin{aligned} \text{var}\left(\Delta^1\right) - \text{var}\left(D^0\right) \leq 2\left[\text{var}\left(X\right) + \text{var}\left(C^p\right) + \text{cov}\left(I^p, C^p\right)\right] \\ - E\left[\left\{\left(D^0 - X - C^p\right)^-\right\}^2\right] \end{aligned} \tag{4.34}$$

Similarly with $D^1 = I^r + u + C^r$, take the variance, and apply the upper bound for var (I^r) from Lemma 4.12 to obtain

$$\text{var}\left(D^1\right) - \text{var}\left(\Delta^1\right) \leq 2\left[\text{var}\left(C^r\right) + \text{cov}\left(I^r, C^r\right)\right] - E\left[\left\{\left(\Delta^1 - u - C^r\right)^-\right\}^2\right] \tag{4.35}$$

Sum inequalities (4.34) and (4.35) and we are done. □

Similar to the case with no cross traffic, the last two terms on the RHS of (4.33) is large if D^0 has a large probability of taking small values, and as such, the observation from the end of Section 4.5.2 carry over and it is reasonable to expect that even with cross traffic, the change in packet clustering, var $\left(D^1\right)$ − var $\left(D^0\right)$ to decrease as the input packet stream is getting more clustered or as the input data rate increases.

4.8 Minimizing Jitter

We can apply the result that we just obtained to show how a traffic engineer could minimize jitter. To start, there exists several definitions for jitter and the one we are going to look at is interpacket delay variation (IPDV) as defined in RFC5481 [78]. More specifically, we are investigating the variance of IPDV, which we will refer to interchangeably with jitter. In the notation of this paper, IPDV is defined as $D_i^n - D_i^0$ and thus for homogeneous input traffic, its variance is simply var $\left(D_i^n\right)$ and we could apply the results that we have so far in the context of jitter.

From the discussion that we just had, we know that jitter would grow slower or even decrease if the interpacket gap in between packets are smaller. The phenomena of decreasing jitter with smaller interpacket gap has been observed in [31], albeit the definition of jitter used is different. Thus, to control jitter, we need to decrease interpacket gap, which could be achieved in three ways. The first is by sending at a higher data rate, which will increase the end-to-end delay as a tradeoff. Alternatively, we could send with smaller packet sizes, though this means we have to send more packets and thus more data overhead in terms of packet headers. Finally, we could send the traffic via a dedicated, rate-limited tunnel through the network though setting up such a tunnel is expensive. Each method of controlling jitter comes with its own tradeoff, and we leave a more thorough investigation for future work.

4.9 Experiments

4.9.1 Experiment Setup and Simulation

To validate our model, we deploy a SoNIC board on a Dell T7500 workstation, and use a Cisco Catalyst 6500 router. The workstation is equipped with two 2.93 GHz six-core Xeon 5670 processors, and with 12 GB RAM, 6 GB connected to each of the processors. Two Myricom SFP+ LR transceivers are plugged into the SoNIC board for wire connections. The Cisco router uses Cisco Supervisor Engine 720 for the control plane, and four-port 10 Gigabit Ethernet Fiber module (WS-X6704-10GE) for the data plane. The router is configured to run in default L2 forwarding mode. We connect two ports of the SoNIC board to two 10 GbE ports of the router via LC/SC optical fibers. Then, we use one port of SoNIC to generate packets, and the other port of SoNIC to capture packets routed from the router.

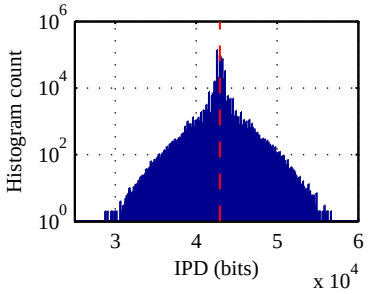
We extend the functionality of SoNIC to mimic a multi-hop routing environment. In particular, after capturing and recording the IPD from the output port, we use it to generate a new stream of packets with IPDs identical to the recorded data. This new packet stream is then sent to the input port and the process can be repeated any number of times. This is possible because SoNIC can capture what is sent including the size of packets along with headers and the number of idle characters and bits between any two packets. Therefore, SoNIC can easily reconstruct an identical packet stream with captured information. We use this functionality to validate our model and scale the experiment up to 20 hops or more.

A good way to check the validity of the model is to simulate the model and compare it with experimental data with the same parameters. We code up the model in Matlab. For accurate modeling, ideally we would need to figure out the distribution of the processing time, X for the Cisco 6500 router that we are simulating. However, it turns out that the variance of processing time, var (X), alone is sufficient to run the simulation. What we observe is that if we are only concerned about packet clustering, then the underlying distribution does not matter too much as long as its variance is the same as var (X), as estimated using Lemma 4.4. The simulation in this section are thus all run with the processing time taking on an exponential distribution with the same variance as var (X).

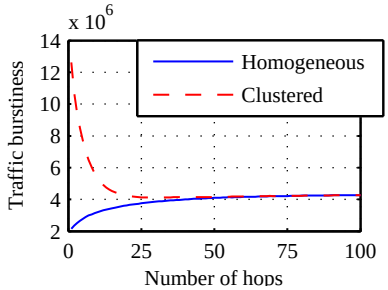
4.9.2 Validation

All the experiments in this subsection are performed with the target traffic having at least 1 million packets. The packet size of the cross traffic is assumed to follow a log-normal distribution [34], while its variance could be estimated from Internet packet size study, e.g. [96], to be in the order of 10^6 bits^2 or higher. In all the experiments, packet clustering is calculated by finding the variance of the interpacket delays of the target traffic. For experiment with multiple hops, twenty hops are usually enough for practical purposes but we sometimes extend to 100 hops when we are concerned with the asymptotic behavior.

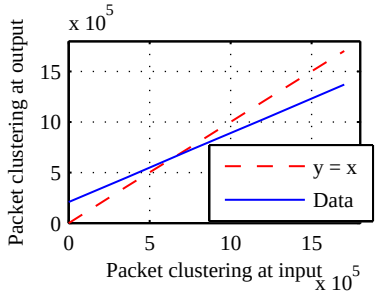
Experiment 1: Negative correlation of adjacent interpacket delays (Theorem 4.2) The correlation coefficient is computed using 1 million IPDs obtained from experiment for various setups. The results are summarized in Table 4.2. We can see that the computed values are close to the value of -0.5. The 3 setups where



(a) IPD histogram for a target traffic of 520B 1G and a cross traffic of 1.2G and a packet size following a log-normal distribution with a mean of 520 bytes and variance of 5.12 $\times$ 10^6 bits^2.



(b) How packet clustering evolves over multiple routers for clustered vs. homogenous packet stream. Both packet streams are 72B 3G.



(c) How packet clustering changes from input to output. The target traffic has a data rate of 3G, packet size of 800B and packet clustering controlled by the parameter of a geometric random variable representing the number of packets clustered together.

Figure 4.10: Results for Experiments 2 and 3

the correlation coefficients are far from -0.5 are from the small interpacket delay region. Even then, they confirm our intuition that the correlation coefficient for adjacent IPD is negative.

Experiment 2: Symmetry in the observed distribution of output IPD for large interpacket gap (Theorem 4.2 and Theorem 4.3) We already know that this is confirmed by figure 4.1a for the no cross traffic case. For the cross traffic case,

its observation is mentioned in RFC5481 [78]. We show here an experiment with cross traffic demonstrating the symmetry in Figure 4.10a. The correlation coefficient of adjacent interpacket delays is calculated to be -0.4944.

Experiment 3: Packet clustering could decrease after passing through one router (Corollary 4.7 and Theorem 4.11) We verify that for a very clustered input packet stream, packet clustering decreases after the packet stream passes through the router. Without cross traffic, the result is shown in Figure 4.10b. The clustered packet stream has 10 packets clustered together with minimal IPG and one huge gap before the next cluster. As a comparison, we also plotted the change in packet clustering for a homogenous packet stream with the same packet size and data rate. Notice that as the number of hops increases, packet clustering varies in an increasing and concave manner for the homogeneous packet stream, and decreasing and convex for the clustered packet stream. This agrees with the discussion in Section 4.6.2.

With cross traffic, the result is shown in Figure 4.10c. The red dashed line is the $y = x$ line while the blue line represents the collected data on packet clustering at the output. The height difference between the blue line and the red line represents the change in packet clustering. When the blue line is above

Table 4.2: Correlation coefficient values under various data rates and packet sizes. The 3 starred values are setups that belong to the small interpacket delay region (see Figure 4.4).

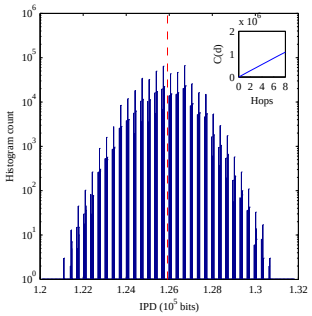
	1526-byte packet	72-byte packet
1 Gbps	-0.4942	-0.5022
3 Gbps	-0.5494	-0.8673*
6 Gbps	-0.4740	-0.1788*
9 Gbps	-0.4931	-0.6924*

the red line, it means that change in packet clustering from input to output is positive and vice versa. We see from the figure that as packet clustering at the input increases, the change in packet clustering decreases and eventually goes negative.

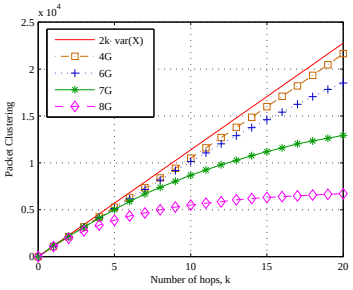
Experiment 4: IPD properties for the multi-hop, large IPG setup (Theorem 4.8) For the large interpacket gap region, a 1526B 1G homogeneous packet stream goes through 8 identical routers and the IPD is recorded. The IPD histogram is observed to be symmetrical and packet clustering grows linearly. In addition, the correlation coefficient between adjacent IPD is calculated as -0.4858, which is close to the theoretical value of -0.5.

Experiment 5: How packet clustering evolves for different data rates with increasing number of hops (Section 4.6.2) Figure 4.11b shows experimental data for fixed packet size but varying data rates. The 4, 6 and 7 Gbps setup could be thought of as representing medium IPD, and 8 Gbps the small IPD. For the first few hops, packet clustering evolves almost linearly for all data rates. As the number of hops increases, the increment decreases at a faster rate for the higher rate. The figure tells us that for increasing data rates, packet clustering increases at a decreasing rate, in agreement with the discussion in Section 4.6.2.

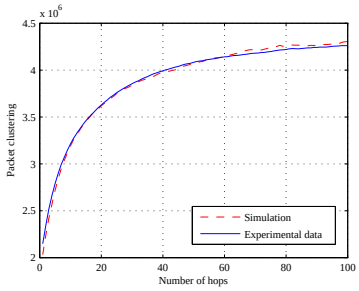
Experiment 6: Validation via simulation Figure 4.11c shows how packet clustering changes as a homogeneous traffic passes through multiple routers for experimental and simulated data. Note that though we have used an exponential distribution to represent X, the simulated values for the model agrees closely with the experimental data even after 100 hops.



(a) Interpacket delay after 8 hops for homogeneous input traffic of 1526-byte packet and 1 Gbps data rate. Inset shows the linear growth of traffic burstiness.



(b) How packet clustering evolves as a homogenous packet stream of 520-byte packets passes through different number of routers for various data rates.



(c) Simulation vs. experiment. The input homogeneous traffic has 72-byte packet and 3 Gbps data rate.

Figure 4.11: Results of Experiments 4 - 6

4.9.3 Factors Affecting Inherent Variation

4.9.3.1 Packet Size

We test the effect of packet size by sending streams of homogeneous traffic with fixed packet sizes of $94, 158, 222, \ldots, 1502$ bits through a router. We then measure the sample variance of the IPD at the output of the router. To guarantee that this is an accurate description of the inherent variation of the router according to Lemma 4.4, we need the packet streams to be spaced sufficiently apart and in this case, we have an interpacket gap of 113340 bits for all streams. The result

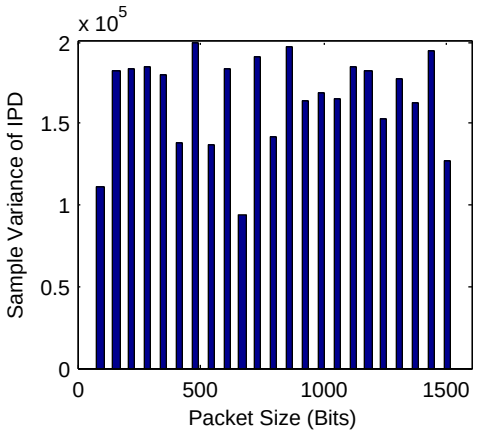


Figure 4.12: Sample variance of IPD for various packet sizes. Each bar is spaced apart by 64 bits.

is shown in Figure 4.12. There is no obvious relationship between packet size and inherent variation and the values vary significantly as the highest value is more than twice the lowest value. The high variation is not an artifact of insufficient sample size, as we repeated the experiments multiple times to get similar values.

4.9.3.2 Forwarding Table Lookup

We first populate the forwarding tables fully with entries, all of which direct packets to the same output port. Then using the idea from the proof of Theorem 4.3, we construct a 1518B 1G stream of packets where the even packets contain destinations that are chosen to hit a random entry of the forwarding table while the odd packets all contain the same destination which is not in the forwarding table. The odd packets will be broadcasted to all output ports. We record the packet stream at the output port, split it into two streams of even and odd packets, and measure the sample variance of IPD of each stream.

The calculated sample variance is 1.53×10^5 seconds^2 and 1.36×10^5 seconds^2 for the even and odd packet stream respectively, which shows that forwarding table lookup do cause inherent variation.

4.9.3.3 Clock Drift

Clock drift happens when two clocks are not running at the same frequency and in this case, the two clocks refer to the routers and SoNIC. We did not actually set out to test for this phenomenon but instead, we notice the difference in packet timing as shown in Figure 4.13 that could not be explained otherwise. To measure the timing difference, we use the recorded output stream and choose a bit on the stream as the reference bit. We measure the difference in position in between the reference bit and the first packet, then we move the position of the reference bit forward by the IPD constant and then measure the difference in position in between it and the second packet. We repeat the same procedure for all the packets. The expectation is that the figure would show fluctuations reflecting the inherent variation around a horizontal line. Instead, other than the fluctuations, there is a variation in the timing difference that is on a longer timescale compared to the fluctuations.

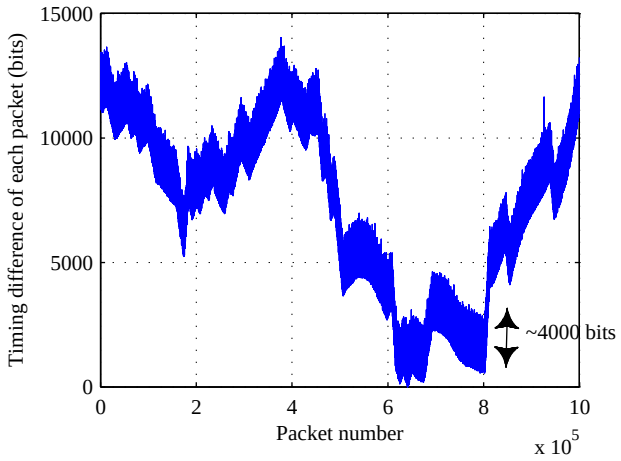


Figure 4.13: Figure shows the difference between packet timing at Cisco 6500 output and input for homogeneous traffic with 1500-byte packets and 1 Gbps data rate. All the values have been shifted by a constant to illustrate the magnitude of the changing latency difference. The band height of approximately 4000 bits is in agreement with the spread of values in Figure 4.1a.

CHAPTER 5

FUTURE WORK

This thesis has touched on various aspects related to traffic engineering and application requirements. We have covered how the proper feedback could lead to network-user optimality, how timing information could be used to achieve fast congestion-free routing reconfiguration, and how inherent variation in routers could induce packet clustering,. The work is not done though, as they remain several interesting extensions.

5.1 Network-User Challenges

While we have shown that it is possible to achieve stability and social optimality with the right feedback, they are predicated on a few modeling assumptions that deserve further investigation. First, we have assumed that traffic could be split arbitrarily across multiple paths. This is achievable if the underlying routing protocol is path-based such as MPLS [89]. But if the routing protocols are link-state protocols such as OSPF and IS-IS, then due to the NP-hard link-weight setting problem, it is not always possible for the traffic engineer to find the correct link weight to route the traffic optimally within limited time. However, our results hold as long as the traffic engineer finds a solution that improves on the current objective strictly. This is likely doable with recent development in link-state routing as shown in [109, 75] but the question remains that whether it is doable for the current link-state protocols that are widely used.

Second, we have also assumed that the updates are instantaneous. In practice, for the traffic engineer, there is some delay between obtaining the mea-

surement of the offered traffic and updating the traffic split in the routers. The offered traffic may have changed slightly during the delay. The bigger problem though, is that a user may be receiving feedback that are no longer up-to-date as the traffic engineer changes the traffic split. Additional conditions are likely needed to prove stability and optimality in this scenario.

5.2 Even Faster Reconfiguration

We formulated an optimization problem incorporating timing information on propagation delay and uncertainties to find fast congestion-free updates when the routes need to be reconfigured. We no longer have to assume a worst-case order-oblivious scenario but instead could determine which scenario we need to consider to prevent congestion. We can calculate the time required for each update to fully propagate through the network. The key problem is the associated optimization problem is a mixed-integer linear program and is NP-hard to solve. An obvious extension is to find approximation algorithms with good performance guarantee.

We have shown it is possible to speed up the process by performing updates before the previous update has propagated through. We could extend the idea to include not just the previous step, but all previous steps. The problem would be more complicated as the possible updates evolves over time and would require a state space to keep track, but the update time would be even faster.

Our optimization problem deals with uncertainty, and while we achieve our goal of finding better updates compared to [51] by incorporating timing information, there exist other tools and framework that could be applicable to

our problem at hand. Most notably, the framework of robust optimization [10] caters specifically to optimization and decision making under uncertainty, and it would be interesting to see what a formulation under such a framework would bring us compared to our current approach.

5.3 Router and Traffic Modeling

Our router model has formed a framework for understanding how inherent variation in a router affects the input-output characteristic of a router. There are plenty of interesting research directions that we could pursue with the model as a starting point. On the practical side, we could delve into the workings of a router, try to understand how inherent variation comes about and build a practical router model in the same spirit as [26] but which incorporates inherent variation as an important parameter. On the application side, as mentioned before, we could investigate deeper into the tradeoffs between the various methods of controlling jitter by minimizing interpacket gap. We could also look at bandwidth estimation using packet-train dispersion [67] and see if our analysis could help in improving the accuracy of existing bandwidth estimation methods.

BIBLIOGRAPHY

[1] Cisco Catalyst 6500 white paper. http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/prod_white_paper0900aecd80673385.html.

[2] IEEE Standard 802.3-2008. http://standards.ieee.org/about/get/802/802.3.html.

[3] Iperf. https://code.google.com/p/iperf/.

[4] Mininet. http://mininet.org/.

[5] Netrunner. http://www.netrunner-os.com/.

[6] Open Networking Foundation. https://www.opennetworking.org/.

[7] Pox wiki. https://openflow.stanford.edu/display/ONL/POX+Wiki.

[8] Summary of the Amazon EC2 and Amazon RDS service disruption in the US East region. http://aws.amazon.com/message/65648/.

[9] ANDERSON, E., AND ANDERSON, T. On the stability of adaptive routing in the presence of congestion control. In *IEEE INFOCOM* (2003).

[10] BEN-TAL, A., EL GHAOUI, L., AND NEMIROVSKI, A. *Robust optimization*. Princeton University Press, 2009.

[11] BERAN, J. *Statistics for long-memory processes*, vol. 61. Chapman & Hall/CRC, 1994.

[12] BERTSEKAS, D. *Nonlinear programming*. Athena Scientific, 1999.

[13] BERTSEKAS, D., AND GALLAGER, R. *Data networks*. Prentice Hall, 1992.

[14] BERTSEKAS, D., AND TSITSIKLIS, J. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[15] BLACK, U. D. *IP routing protocols: RIP, OSPF, BGP, PNNI, and Cisco routing protocols*. Prentice Hall Professional, 2000.

[16] BLAKE, S., BLACK, D., CARLSON, M., DAVIES, E., WANG, Z., AND WEISS, W. RFC 2475: An architecture for differentiated services.

[17] BOLOT, J.-C. End-to-end packet delay and loss behavior in the internet. *ACM SIGCOMM Computer Communication Review 23*, 4 (1993), 289–298.

[18] BOVY, C., MERTODIMEDJO, H., HOOGHIEMSTRA, G., UIJTERWAAL, H., AND VAN MIEGHEM, P. Analysis of end-to-end delay measurements in internet. In *Passive and Active Measurements* (2002).

[19] BOYD, S., AND VANDENBERGHE, L. *Convex Optimization*. Cambridge University Press, 2004.

[20] BRAKMO, L. S., AND PETERSON, L. TCP Vegas: End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communications 13*, 8 (1995), 1465–1480.

[21] BROIDO, A., KING, R., NEMETH, E., AND CLAFFY, K. Radon spectroscopy of inter-packet delay. In *High Speed Networks Workshop* (2003).

[22] CALLON, R. Use of OSI IS-IS for routing in TCP/IP and dual environments.

[23] CANTOR, D. G., AND GERLA, M. Optimal routing in a packet switched computer network. *IEEE Transactions on Computers C-23* (1974), 1062–1069.

[24] CHAO, H. J., AND LIU, B. *High Performance Switches and Routers*. Wiley, 2007.

[25] CHAU, C. K., AND SIM, K. M. The price of anarchy for non-atomic congestion games with symmetric cost maps and elastic demands. *Operations Research Letters 31*, 5 (2003), 327 – 334.

[26] CHERTOV, R., AND FAHMY, S. Forwarding devices: From measurements to simulations. *ACM Transactions on Modeling and Computer Simulation 21*, 2 (2011), 12.

[27] CHIANG, M., LOW, S., CALDERBANK, A., AND DOYLE, J. Layering as optimization decomposition: A mathematical theory of network architectures. *Proc. IEEE 95*, 1 (Jan. 2007), 255 –312.

[28] COLE, R., DODIS, Y., AND ROUGHGARDEN, T. Bottleneck links, variable demand and the tragedy of the commons. In *ACM-SIAM Symposium on Discrete Algorithms* (2006).

[29] COVER, T., AND THOMAS, J. *Elements of information theory*. Wiley-Interscience, 2006.

[30] CROVELLA, M., AND BESTAVROS, A. Self-similarity in world wide web traffic: Evidence and possible causes. *IEEE/ACM Transactions on Networking 5*, 6 (1997), 835–846.

[31] DAHMOUNI, H., GIRARD, A., AND SANSÒ, B. An analytical model for jitter in IP networks. *Annals of Telecommunications 67*, 1-2 (2012), 81–90.

[32] DE VITO, L., RAPUANO, S., AND TOMACIELLO, L. One-way delay measurement: State of the art. *IEEE Transactions on Instrumentation and Measurement 57*, 12 (2008), 2742–2750.

[33] DIPALANTINO, D., AND JOHARI, R. Traffic engineering vs. content distribution: A game theoretic perspective. In *IEEE INFOCOM* (2009).

[34] DOWNEY, A. B. The structural cause of file size distributions. In *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (2001), pp. 361–370.

[35] FEAMSTER, N., REXFORD, J., AND ZEGURA, E. The road to SDN. *Queue 11*, 12 (2013), 20.

[36] FORTZ, B., REXFORD, J., AND THORUP, M. Traffic engineering with traditional ip routing protocols. *IEEE Communications Magazine 40*, 10 (Oct 2002), 118 – 124.

[37] FORTZ, B., AND THORUP, M. Internet traffic engineering by optimizing OSPF weights. In *IEEE INFOCOM* (2000).

[38] FORTZ, B., AND THORUP, M. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications 20*, 4 (2002).

[39] FORTZ, B., AND THORUP, M. Increasing internet capacity using local search. *Computational Optimization and Applications 29* (2004), 13–48.

[40] FRALEIGH, C., MOON, S., LYLES, B., COTTON, C., KHAN, M., MOLL, D., ROCKELL, R., SEELY, T., AND DIOT, S. Packet-level traffic measurements from the sprint ip backbone. *IEEE Network 17*, 6 (2003), 6–16.

[41] FRANCOIS, P., AND BONAVENTURE, O. Avoiding transient loops during the convergence of link-state routing protocols. *IEEE/ACM Transactions on Networking 15*, 6 (2007), 1280–1292.

[42] FRATTA, L., GERLA, M., AND KLEINROCK, L. The flow deviation method: An approach to store-and-forward communication network design. *Networks 3*, 2 (1973), 97–133.

[43] FREEDMAN, D., MARIAN, T., LEE, J. H., BIRMAN, K., WEATHERSPOON, H., AND XU, C. Exact temporal characterization of 10 Gbps optical wide-area network. In *ACM Internet Measurement Conference* (2010).

[44] FUDENBERG, D., AND TIROLE, J. *Game theory*. MIT Press, 1991.

[45] GALLAGER, R. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications 25*, 1 (Jan 1977), 73 – 85.

[46] HE, J., BRESLER, M., CHIANG, M., AND REXFORD, J. Towards robust multi-layer traffic engineering: Optimization of congestion control and routing. *IEEE Journal on Selected Areas in Communications 25*, 5 (June 2007), 868–880.

[47] HE, J., CHIANG, M., AND REXFORD, J. TCP/IP interaction based on congestion price: Stability and optimality. In *IEEE International Conference on Communications* (June 2006).

[48] HEART, F. E., KAHN, R. E., ORNSTEIN, S., CROWTHER, W., AND WALDEN, D. C. The interface message processor for the ARPA computer network. In *Proceedings of the May 5-7, 1970, Spring Joint Computer Conference* (1970), AFIPS '70 (Spring), pp. 551–567.

[49] HEDRICK, C. RFC 1058: Routing information protocol.

[50] HOHN, N., VEITCH, D., PAPAGIANNAKI, K., AND DIOT, C. Bridging router performance and queuing theory. In *ACM SIGMETRICS* (2004).

[51] HONG, C.-Y., KANDULA, S., MAHAJAN, R., ZHANG, M., GILL, V., NANDURI, M., AND WATTENHOFER, R. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM* (2013).

[52] JIANG, H., AND DOVROLIS, C. Why is the internet traffic bursty in short time scales. In *ACM SIGMETRICS* (2005).

[53] JIANG, L., AND WALRAND, J. *Scheduling and Congestion Control for Wireless and Processing Networks*. Morgan & Claypool, 2010.

[54] JOHARI, R., AND TSITSIKLIS, J. A scalable network resource allocation mechanism with bounded efficiency loss. *IEEE Journal on Selected Areas in Communications 24*, 5 (May 2006), 992 – 999.

[55] KANDULA, S., KATABI, D., SINHA, S., AND BERGER, A. Dynamic load balancing without packet reordering. *ACM SIGCOMM Computer Communication Review 37*, 2 (2007), 51–62.

[56] KATTA, N. P., REXFORD, J., AND WALKER, D. Incremental consistent updates. In *ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking* (2013).

[57] KELLY, F. P. *Reversibility and Stochastic Networks*. Cambridge University Press, 2011.

[58] KELLY, F. P., MAULLOO, A. K., AND TAN, D. K. H. Rate control for communication networks: Shadow prices, proportional fairness and stability. *Journal of the Operational Research Society 49*, 3 (1998), pp. 237–252.

[59] KINGMAN, J. F. C. Some inequalities for the queue GI/G/1. *Biometrika 49*, 3/4 (1962), 315–324.

[60] KINGMAN, J. F. C. Inequalities in the theory of queues. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 32* (1970).

[61] KUROSE, J. On computing per-session performance bounds in high-speed multi-hop computer networks. *ACM SIGMETRICS Performance Evaluation Review 20*, 1 (June 1992), 128–139.

[62] LEE, K., WANG, H., AND WEATHERSPOON, H. SoNIC: Precise Realtime Software Access and Control of Wired Networks. In *USENIX Symposium on Networked Systerms Design and Implementation* (2013).

[63] LELAND, W. E., TAQQU, M. S., WILLINGER, W., AND WILSON, D. V. On the self-similar nature of ethernet traffic. In *ACM SIGCOMM Computer Communication Review* (1993), vol. 23, pp. 183–193.

[64] LIM, C. L., LEE, K. S., WANG, H., WEATHERSPOON, H., AND TANG, A. Packet clustering introduced by routers: Modeling, analysis and experiments. In *Conference on Information Sciences and Systems* (2014).

[65] LIM, C. L., AND TANG, A. Traffic engineering with elastic traffic. In *IEEE GLOBECOM* (2013).

[66] LINDLEY, D. The theory of queues with a single server. In *Math. Proc. Carmbridge* (1952), vol. 48, Cambridge University Press, pp. 277–289.

[67] LIU, X., RAVINDRAN, K., AND LOGUINOV, D. A queueing-theoretic foundation of available bandwidth estimation: Single-hop analysis. *IEEE/ACM Transactions on Networking 15*, 4 (aug. 2007), 918 –931.

[68] LOW, S. A duality model of TCP and queue management algorithms. *IEEE/ACM Transactions on Networking 11*, 4 (Aug. 2003), 525 – 536.

[69] LOW, S., AND D.E., L. Optimization flow control, I: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking 7*, 6 (1999), 861–874.

[70] LUO, Y., AND ANSARI, N. Bandwidth allocation for multiservice access on epons. *IEEE Communications Magazine 43*, 2 (2005), S16–S21.

[71] MALKIN, G. RFC 2453: RIP version 2.

[72] MARIAN, T., FREEDMAN, D., BIRMAN, K., AND WEATHERSPOON, H. Empirical characterization of uncongested optical lambda networks and 10GbE commodity endpoints. In *IEEE/IFIP International Conference on Dependable Systems and Networks* (2010).

[73] MARTIN J. BECKMANN, C. B. MCGUIRE, C. B. W. *Studies in the Economics of Transportation*. Yale University Press, 1956.

[74] MEYER, M., AND VASSEUR, J. RFC 5712: MPLS traffic engineering soft preemption.

[75] MICHAEL, N., TANG, A., AND XU, D. Optimal link-state hop-by-hop routing. In *IEEE International Conference on Network Protocols* (2013).

[76] MO, J., AND WALRAND, J. Fair end-to-end window-based congestion control. *IEEE/ACM Transactions on Networking 8*, 5 (Oct 2000), 556–567.

[77] MONDERER, D., AND SHAPLEY, L. S. Potential games. *Games and Economic Behavior 14*, 1 (1996), 124 – 143.

[78] MORTON, A., AND CLAISE, B. RFC 5481: Packet delay variation applicability statement.

[79] MOY, J. RFC 2328: OSPF version 2.

[80] NEMHAUSER, G. L., AND WOLSEY, L. A. *Integer and combinatorial optimization*, vol. 18. Wiley New York, 1988.

[81] NUCCI, A., BHATTACHARYYA, S., TAFT, N., AND DIOT, C. IGP link weight assignment for operational tier-1 backbones. *IEEE/ACM Transactions on Networking 15*, 4 (2007), 789–802.

[82] PAGANINI, F., AND MALLADA, E. A unified approach to congestion control and node-based multipath routing. *IEEE/ACM Transactions on Networking 17*, 5 (Oct. 2009), 1413 –1426.

[83] PARK, K., KIM, G., AND CROVELLA, M. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *IEEE International Conference on Network Protocols* (1996).

[84] PATHAK, A., ZHANG, M., HU, Y. C., MAHAJAN, R., AND MALTZ, D. Latency inflation with MPLS-based traffic engineering. In *ACM Internet Measurement Conference* (2011).

[85] PAXSON, V. End-to-end internet packet dynamics. *IEEE/ACM Transactions on Networking 7*, 3 (1999), 277–292.

[86] PRIVALOV, A., AND SOHRABY, K. Per-stream jitter analysis in cbr atm multiplexors. *IEEE/ACM Transactions on Networking 6*, 2 (1998), 141–149.

[87] RAZA, S., ZHU, Y., AND CHUAH, C.-N. Graceful network state migrations. *IEEE/ACM Transactions on Networking 19*, 4 (2011), 1097–1110.

[88] REITBLATT, M., FOSTER, N., REXFORD, J., SCHLESINGER, C., AND WALKER, D. Abstractions for network update. In *ACM SIGCOMM* (2012).

[89] ROSEN, E. RFC 3031: Multiprotocol label switching architecture.

[90] ROSEN, J. B. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica 33*, 3 (1965), 520–534.

[91] ROSENTHAL, R. W. A class of games possessing pure-strategy nash equilibria. *International Journal of Game Theory 2* (1973), 65–67.

[92] ROUGHGARDEN, T. The price of anarchy is independent of the network topology. In *Journal of Computer and System Sciences* (2002).

[93] ROUGHGARDEN, T., AND TARDOS, E. How bad is selfish routing? *Journal of the ACM 49*, 2 (Mar. 2002), 236–259.

[94] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. RFC 3550 RTP: A transport protocol for real-time applications, 2003. Available: http://tools.ietf.org/html/rfc3550.

[95] SHAKKOTTAI, S., BROWNLEE, N., ET AL. A study of burstiness in TCP flows. In *Passive and Active Network Measurement*. Springer, 2005, pp. 13–26.

[96] SINHA, R., PAPADOPOULOS, C., AND HEIDEMANN, J. Internet packet size distributions: Some observations. *Available: http://netweb. usc. edu/˜ rsinha/pkt-sizes* (2007).

[97] SRIKANT, R. *The mathematics of Internet congestion control*. Birkhäuser, 2004.

[98] SRINIVASAN, K., KAZANDJIEVA, M., AGARWAL, S., AND LEVIS, P. The β-factor: measuring wireless link burstiness. In *ACM Conference on Embedded Networked Sensor Systems* (2008), pp. 29–42.

[99] TANG, A., ANDREW, L., JACOBSSON, K., JOHANSSON, K., HJALMARSSON, H., AND LOW, S. Queue dynamics with window flow control. *IEEE/ACM Transactions on Networking 18*, 5 (2010), 1422–1435.

[100] TANG, A., WANG, J., LOW, S. H., AND CHIANG, M. Equilibrium of heterogeneous congestion control: existence and uniqueness. *IEEE/ACM Transactions on Networking 15*, 4 (Aug 2007), 824–837.

[101] TANG, A., WEI, X., LOW, S. H., AND CHIANG, M. Equilibrium of heterogeneous congestion control: optimality and stability. *IEEE/ACM Transactions on Networking 18*, 3 (Jun 2010), 844–857.

[102] VANBEVER, L., VISSICCHIO, S., PELSSER, C., FRANCOIS, P., AND BONAVENTURE, O. Seamless network-wide IGP migrations. In *ACM SIGCOMM* (2011).

[103] VANBEVER, L., VISSICCHIO, S., PELSSER, C., FRANCOIS, P., AND BONAVENTURE, O. Lossless migrations of link-state IGPs. *IEEE/ACM Transactions on Networking 20*, 6 (2012), 1842–1855.

[104] WALRAND, J. *An introduction to queueing networks*. Prentice Hall, 1988.

[105] WALRAND, J. Entropy in communication and chemical systems. In *International Symposium on Applied Sciences on Biomedical and Communication Technologies* (2008).

[106] WANG, J., LI, L., LOW, S., AND DOYLE, J. Cross-layer optimization in TCP/IP networks. *IEEE/ACM Transactions on Networking 13*, 3 (June 2005), 582 – 595.

[107] WANG, X., REEVES, D., AND WU, S. F. Inter-packet delay based correlation for tracing encrypted connections through stepping stones. In *European Symposium on Research in Computer Security*. 2002.

[108] WILLINGER, W., TAQQU, M., SHERMAN, R., AND WILSON, D. Self-similarity through high-variability: Statistical analysis of ethernet lan traffic at the source level. *IEEE/ACM Transactions on Networking 5*, 1 (1997), 71–86.

[109] XU, D., CHIANG, M., AND REXFORD, J. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *IEEE/ACM Transactions on Networking 19*, 6 (Dec. 2011), 1717 –1730.

[110] YEN, J. Y. Finding the k shortest loopless paths in a network. *Management Science 17*, 11 (1971), 712–716.